Pedro José Marrón   Kamin Whitehouse (Eds.)

Wireless Sensor Networks

8th European Conference, EWSN 2011
Bonn, Germany, February 23-25, 2011
Proceedings

Springer

Volume Editors

Pedro José Marrón
Universität Duisburg-Essen
Fakultät für Ingenieurwissenschaften, Abteilung Informatik
Bismarckstraße 90, 47057 Duisburg, Germany
E-mail: pjmarron@uni-due.de

Kamin Whitehouse
University of Virginia, School of Engineering and Applied Science
Department of Computer Science
151 Engineer's Way, Charlottesville, VA 22904-4740, USA
E-mail: whitehouse@cs.virginia.edu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-19185-5 e-ISBN 978-3-642-19186-2
DOI 10.1007/978-3-642-19186-2
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011920570

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

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

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

Preface

This volume contains the proceedings of EWSN 2011, the 8th European Conference on Wireless Sensor Networks. The conference took place in Bonn, Germany during February 23–25, 2011. The aim of the conference was to discuss the latest research results and developments in the field of wireless sensor networks.

EWSN received a total of 87 paper submissions of which 14 were selected for publication and presentation, yielding an acceptance rate of about 16%. Paper submissions were received from 29 different countries in all parts of the world. EWSN adopted a double-blind review process, where the identities of the paper authors were also withheld from the reviewers. The selection process involved around 250 reviews with all papers being evaluated by at least three independent reviewers. In addition, the reviews where discussed by the Technical Program Committee in a virtual meeting after collecting all reviews and prior to making final decisions. The final program covered a wide range of topics which were grouped into five sessions: Routing and Mobility, Optimization Techniques, MAC Protocols, Algorithms for Wireless Sensor Networks, and Systems and Abstractions. It included theoretical and analytical approaches, together with empirical research and protocol/system design and implementation.

The conference included a keynote by Mani Srivastava with the title "System Issues in Wireless Sensor Networks," a demo and poster session, co-chaired by Luca Mottola and Daniel Minder, for which separate proceedings are available, and an industrial demo session, co-chaired by Herman Tuininga and Siebren de Vries, where companies working in the area of wireless sensor networks had the chance to exhibit their products throughout the conference. The conference also included a tutorial on "Machine Learning for Wireless Sensor Networks" by Anna Förster and a tutorial on "TeenyLIME" by Amy Murphy.

We would like to thank everyone who contributed to EWSN 2011. In particular, we would like to thank the Technical Program Committee for their reviews and input in forming the program. We also would like to thank the local administration at the University of Bonn for their help with the conference planing and last, but certainly not least, our sponsors: Networked Embedded Systems Group at the University of Duisburg-Essen (Gold Sponsor), CONET Network of Excellence (Gold Sponsor), Boeing (Bronze Sponsor), and Libelium (Bronze Sponsor).

February 2011

Pedro José Marrón
Kamin Whitehouse

Organization

Michele Rossi, University of Padova, Italy
Antonio Ruzzelli, UCD Dublin, Ireland
Andreas Savvides, Yale University, USA
Cormac Sreenan, Cork University, Ireland
Nigamanth Sridhar, Cleveland State University, USA
Andreas Terzis, Johns Hopkins University, USA
Niwat Thepvilojanapong, Mie University, Japan
Thiemo Voigt, Swedish Institute of Technology, Sweden
Andreas Willig, Technical University of Berlin, Germany
Adam Wolisz, Technical University of Berlin, Germany

Poster and Demo Chairs:
Daniel Minder, University of Duisburg-Essen
Luca Mottola, Swedish Institute of Computer Science, Sweden

Industrial Demo Committee:
Pedro José Marrón, University of Duisburg-Essen and Fraunhofer IAIS, Germany
Herman Tuininga, CEO and owner of SallandElectronics–Zwolle, Netherlands
Siebren de Vries, CEO and owner of Chess bv–Haarlem, Netherlands

Sponsors:
Networked Embedded Systems Group at the University of Duisburg-Essen (Gold)
CONET Network of Excellence (Gold)
Boeing (Bronze)
Libelium (Bronze)

Table of Contents

Prediction Accuracy of Link-Quality Estimators

Christian Renner, Sebastian Ernst, Christoph Weyer, and Volker Turau

Hamburg University of Technology, Hamburg, Germany
{christian.renner,c.weyer,turau}@tu-harburg.de

Abstract. The accuracy of link-quality estimators (LQE) is mission-critical in many application scenarios in wireless sensor networks (WSN), since the link-quality metric is used for routing decisions or neighborhood formation. Link-quality estimation must offer validity for different timescales. Existing LQEs describe and approximate the current quality in a single value only. This method leads to a limited accuracy and expressiveness about the presumed future behavior of a link. The LQE developed in this paper incorporates four quality metrics that give a holistic assessment of the link and its dynamic behavior; therefore, this research is an important step to achieving a higher prediction accuracy including knowledge about the short- and long-term behavior.

1 Introduction

For most algorithms in wireless sensor networks (WSN) it is essential that each node has thorough knowledge about its direct neighbors. This information is collected and provided by neighborhood management protocols and is used, e.g., for routing decisions, group formation, or data sharing. The dynamic behavior over time of the wireless channel and the missing correlation between adjacency and possibility of communication—due to obstacles and multi-path propagation—render the definition of the neighborhood of a node a non-trivial task. One important criterion used by neighbourhood management protocols to determine the importance of a node is the quality of the communication between nodes, which in turn is provided by a link-quality estimator (LQE). Depending on the intended application, either the short- or long-term qualities of a link or a combination of both is preferred for choosing an appropriate node in the vicinity [2]. In the past many different approaches were investigated [14,6,4,7,2,11,3,1] or are currently used, e.g., in TinyOS [10,8].

In principle a LQE measures the quality based on logical (e.g., packet success rate) or physical (e.g., received signal strength) metrics. Newer proposals use a combination of these in order to improve the accuracy of the prediction [1]. However, common to these LQEs is that the measured quality is squeezed into a single value—e.g., a moving average—due to memory restrictions and for easier comparability. In doing so, the value represents only a snap-shot of the plain link-quality at a specific point in time without any additional information about variation and the current trend of the long- or short-term behavior in the past. The expressiveness of such a single-value metric is limited: This procedure is

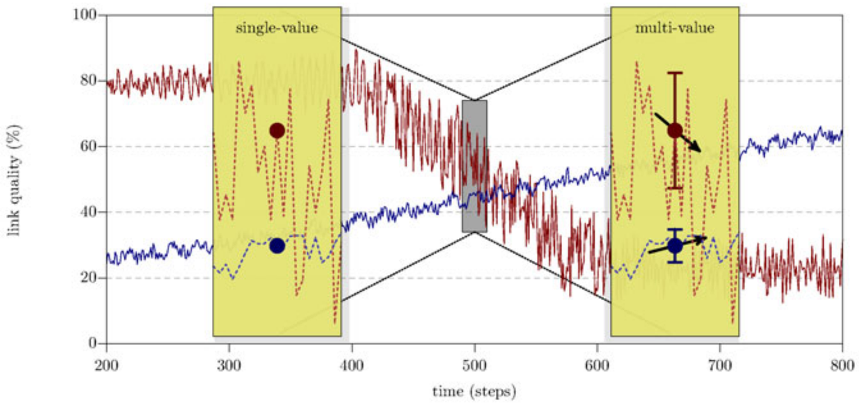


Fig. 1. Link-quality estimation of a deteriorating and an improving link with a single-value (left) and statistical multi-value (right) technique

comparable to a stock market, where a stock is described only by its current value or an average of its last values. Based on this limited information it is a game of luck to decide in which stocks to invest in the future. Furthermore, it is important to know which stock will perform better for a short-term profit and which is suitable for a long-term investment.

The same considerations are valid for judging the quality of a link. Figure 1 visualizes the problem of comparing two link-qualities. With a metric that reflects only the current quality, a comparison is nearly impossible even for a short-term projection. Based on a single-value prediction the deteriorating link appears superior, since its current link-quality is higher. When using multi-value prediction the dynamic behavior of links can also be incorporated into the decision, so that the improving link will be favored: Although its current link-quality is lower, its positive trend and low deviation lead to a better future perspective when compared to the large variation and negative trend of the deteriorating link. Furthermore, all existing LQEs are only capable of providing either a short- or a long-term prediction. Yet, an application often needs different prediction windows, e.g., a long-term prediction for forming a cluster and a short-term one for selecting the eligible next routing hop.

In this paper we present a LQE that effectively calculates for each link four quality characteristics: short- and long-term quality, variation, and an indicator of the current trend. This LQE is a general approach that can be easily adopted to various sources of link-quality metrics. Intensive simulations and experiments are undertaken in order to proof the advantages of the newly developed approach over existing ones. We compare and analyze the ability of link-quality tracking of modern and well-known LQEs with our new approach. In this context, we identify problems and benefits of the different estimators, and we also determine the correlation and average error with respect to empirical ground-truth measures. Moreover, we analyze the ability of the LQEs to select reliable links, which is commonly needed for routing decisions and cluster formation. The paper concludes with a discussion of the results and intended future research activities.

2 Link-Quality Estimation Techniques

Estimating link-qualities in WSNs is a nontrivial task. Many existing solutions for other wireless networks are not feasible for WSNs. Particularly, the lack of infra-structure, limited memory, energy constraints, and low-cost transceivers complicate possible solutions. Various approaches were proposed in the past to overcome these limitations and to provide a meaningful metric describing the actual link-quality and thus predicting its future behavior. In this section, a brief overview and discussion of general link-quality metrics is carried out, followed by a thorough view on already existing approaches employing them.

2.1 Link-Quality Metrics

Basically, there exist two different categories of link-quality metrics: physical and logical indicators. The former are provided by the radio hardware and are based on the signal strength of a received packet, such as the Received Signal Strength Indication (RSSI), the Link-Quality Indication (LQI), or Signal-to-Noise Ratio (SNR). The logical indicators estimate the link quality by keeping track of message losses. Examples of such metrics are: Packet Success Rate (PSR), Required Number of Packets (RNP) [4], or Expected Transmission Count (ETX) [6] for describing the effort needed to successfully transmit a packet.

Physical Metrics. Using physical metrics has several advantages. The metrics come without any additional costs, since the measurement is performed by the receiver hardware every time a packet is received, or can at least be calculated easily in the case of SNR. Also only a small number of samples is needed to get a first approximation of the link-quality. Additionally, the metrics can be measured by utilizing any traffic on the wireless channel without the need of periodical broadcasts if the application produces enough traffic.

However, several research activities on this topic have shown that the physical metrics are of shortened use [12]. First of all, the metrics are strongly dependent on the receiver hardware—e.g., in our experiments with the Atmel RF230 many links have a RSSI value close to the sensitivity of the transceiver, but provide reliable packet reception at the same time. Secondly, the RSSI and LQI are only available for successfully received packets, but not in the case of packet loss. There exist different attempts to improve the quality of the measurements—e.g., RSSI calibration [5]—but these introduce a higher computation-complexity not suitable for WSNs. Another disadvantage is the expressiveness of these metrics with respect to the application-related perception, i.e., the expected PSR. In [9] Lal et al. show that physical metrics cannot always predict the PSR, especially in case of long-range links with an RSSI near the sensitivity threshold of the radio chip. In those cases the SNR-to-PSR relation is not deducible at all.

Logical Metrics. The advantages of logical metrics are that they do not depend on specific hardware characteristics and correlate directly with an application point of view, i.e., the ratio of successfully transmitted packets. This however leads to the problem that a node needs to track the ratio between the number of successfully received packets versus packet loss in an efficient way. A window-based approach consumes too much memory, and counting received and lost packets does not incorporate that recent events should be weighted higher. Hence, most of the logical metrics are calculated with an Exponentially Weighted Moving Average (EWMA) [14]. In contrast to physical metrics, LQEs based on this method rely on frequent packet transmissions in order to keep the link-quality estimates up-to-date.

Periodic broadcast packets are often used to achieve this goal and are also the most commonly brought-up drawback of these approaches, since they waste energy and occupy the wireless channel. However, the measurements—even of the physical link-quality estimators—are conducted by the receiver, but the information must be available at the sender. To perform the necessary exchange of information for each neighboring node, piggy-backing this data on top of application packets is risky due to increased packet size, especially in dense networks. Moreover, the time of information exchange depends on the network traffic of each node, so that large delays may occur and nodes make decisions based on outdated link-quality data. Thus, a periodic information exchange using dedicated broadcasts is always necessary when link-quality estimation is needed. This mitigates animadversion on the periodical broadcasts.

2.2 Link-Quality Estimators for Sensor Networks

Woo et al. define the Window Mean with EWMA (WMEWMA) [14], one of the first LQEs for estimating the PSR. In a previous work they also investigate and compare WMEWMA with different existing approaches. Thereafter, EWMA-based estimators have been widely adopted in WSNs.

ETX introduced by De Couto et al. [6] tries to estimate the number of transmissions that are necessary to send a packet successfully. The number of received packets within a fixed window is counted and compared to the number of expected packets that are periodically broadcasted by each node. The disadvantage of ETX is that it is only updated at the end of each window. A short window thus leads to a high fluctuation of the ETX metric and a long window to an infrequently updated ETX. Cerpa et al. [4] introduce RNP that incorporates the distribution of losses within the window. They observed that a link with consecutive losses should be rated lower than links with discrete losses. Four-Bit (4B) [7] is based on ETX with several enhancements. They use an EWMA for estimating the ETX and a second one for smoothing the final 4B metric. Additionally, 4B uses additional information from the link and network layer.

The Link Estimation Exchange Protocol (LEEP) is based on 4B and part of the current TinyOS version [8]. LEEP constitutes a layer between the MAC protocol and the application or routing engine, respectively. Whenever a packet is sent, LEEP attaches additional information to that packet: a sequence number, a counter for the number of known neighbors, and the ID and in-bound link quality of these neighbors. If the space (left by the upper layer) in the packet is too small, a round-robin procedure is used. The main disadvantage of LEEP is that in case of consecutive packet losses the link-quality is not updated. In addition to the limited number of possible links, due to memory restrictions, this behavior can result in a full neighborhood table with nodes that no longer exist.

The Adaptive Link Estimator (ALE), introduced by Weyer et al. [13], is an EWMA filter of the measured PSR. Each node sends a beacon per time interval (a so-called round) and records received beacons from other nodes within this interval. At the end of each round, link qualities are recomputed. Upon reception of a beacon from a previously unknown node the associated link-quality is initialized with a value of 50% to achieve a shorter rise-time for new links. The weight of the EWMA is adapted to the quality of the link. For good links ALE uses a higher weight for more stable estimation, while links with a lower quality are estimated in an agile fashion for a faster reaction.

An approach using Kalman Filter Based Link-Quality Estimation is proposed by Senel et al. [11]. They filter the RSSI of successfully received packets with a Kalman filter and subtract the noise floor to obtain an estimation of the SNR. Finally, a PSR is derived by applying a hardware specific SNR-PSR mapping for the transceiver. This approach is very complex and suffers from the restricted correlation between SNR and PSR [9].

The Fuzzy Link-Quality Estimator (F-LQE) is proposed by Baccour et al. [1]. The idea is to combine four different metrics into a single quality indicator using Fuzzy Logic. These four metrics are SNR, PSR, link asymmetry level (ASL), and stability (SF). The SNR is calculated after each successful packet reception by subtracting the power on the channel directly after packet reception from the signal strength of the packet. An EWMA filter is used to obtain the PSR. From the latter, the asymmetry level is obtained by calculating the absolute difference between the unidirectional PSR values of a node pair. The stability is defined as the coefficient-of-variation of the most recent 30 PSR values of a link.

3 Holistic Packet Statistics

All of the approaches presented in Sect. 2.2 have in common that they squeeze link quality into a single value. We argue that there should be a multi-faceted representation of link quality. Hence, we have devised the concept of Holistic Packet Statistics (HoPS). HoPS is tailored to provide detailed information about the static and dynamic behavior of a link using four distinct descriptors of the link quality. Due to the troublesome nature of the expressiveness of SNR values and the corresponding PSR matches—as laid out in Sect. 2.1—we take a different course than recent approaches and focus on the development of a more sophisticated link-quality assessment using logical metrics.

HoPS allows for utilizing enriched link information by granting access to all four link-quality metrics rather than hiding information by compressing them into a single value. Yet, this does not imply that there is no way to combine the values of HoPS into a single link-quality estimate. In this section, we define the four quality metrics of HoPS and present two possible solutions for dynamic link-quality assessment.

3.1 Link-Quality Descriptors

The four link-quality descriptors used by HoPS are measured at the receiver side by monitoring the packet success rate. Thus, a packet should contain a sequence number for the detection of packet loss. Additionally, the metrics should be updated frequently in a constant interval. Many presented LQE implementations (cf. Sect. 2.2) update only after a packet is successfully received; this leads to a wrong estimation if no more packets are received from a node.

Short-term Estimation is realized using a first-order EWMA filter for retrieving the in-bound PSR of a link in the recent past:

$$h_\tau^{\mathrm{ST}} = \alpha \cdot h_{\tau-1}^{\mathrm{ST}} + (1-\alpha) \cdot q_\tau \,. \tag{1}$$

The choice of the coefficient α influences the sensitivity to short-term changes in link quality. In the straight-forward case, q_τ is a binary value indicating whether an expected packet at time τ was received. Another option would be using a windowed mean, if more than one packet is received from the same node within an update interval. This technique is comparable to WMEWMA.

Long-term Estimation is obtained by a second-order EWMA filter, i.e., the values of the first-order estimation in (1) are smoothed by

$$h_\tau^{\mathrm{LT}} = \beta \cdot h_{\tau-1}^{\mathrm{LT}} + (1-\beta) \cdot h_\tau^{\mathrm{ST}} \,. \tag{2}$$

To achieve a strong smoothing effect, choosing β larger than α is advisable.

The dynamics of a link are obtained by means of the lower and upper deviation of the short-term estimation from the long-term estimation. The sum of lower and upper deviation yields the average absolute deviation. Due to space constraints the detailed derivation is omitted. To track the time-variant changes of these deviations and to preserve memory, another EWMA filter is applied:

$$\delta_\tau^+ = \gamma \cdot \delta_{\tau-1}^+ + (1-\gamma) \cdot \varphi\left(h_\tau^{\mathrm{ST}}, h_\tau^{\mathrm{LT}}\right) \,, \tag{3}$$

$$\delta_\tau^- = \gamma \cdot \delta_{\tau-1}^- + (1-\gamma) \cdot \varphi\left(h_\tau^{\mathrm{LT}}, h_\tau^{\mathrm{ST}}\right) \,, \tag{4}$$

$$\text{with} \quad \varphi(x,y) = \begin{cases} x-y, & \text{if } x > y \\ 0 & \text{else} \end{cases} \,. \tag{5}$$

These two values are utilized as follows.

Absolute Deviation Estimation is the estimated average absolute deviation

$$h_\tau^\sigma = \delta_\tau^+ + \delta_\tau^- \,. \tag{6}$$

This value gives an impression of the stability of the link in terms of the variation of h_τ^{ST} around its mean h_τ^{LT}. Its calculation demands less computing power than the recursively determined standard deviation, while not being less expressive.

Trend Estimation is intended to reflect the course that a link is taking:

$$h_\tau^\theta = \delta_\tau^+ - \delta_\tau^- \; . \tag{7}$$

A floating link—i.e., its long-term quality has had no notable changes in the past and is therefore not likely to change in the future—can be identified by values of h^θ close to zero. In contrast, positive values indicate an improving link, whereas negative values expose a deteriorating link. The absolute value of the trend indicates the slope of the current trend.

3.2 Theory in Praxis

At this point, we give a brief introduction to the interpretation and usage of the four link-quality metrics of HoPS. There are many possible solutions for the utilization of the four link-quality indicators of HoPS. For instance, a routing protocol could favor a less varying link over one with the same long-term quality. Another possibility would be to use the variation and trend in order to compute a lower bound link-quality with a given confidence. Hence, there is no silver bullet for an enriched employment of these values. However, we want to present two examples of merging the four HoPS ingredients into a prediction value.

The first approach is a dynamically adjusted link-quality estimator:

$$H_\tau^{\text{dyn}} = h_\tau^{\text{LT}} + \frac{\left|h_\tau^\theta\right|}{h_\tau^\sigma} \cdot \left(h_\tau^{\text{ST}} - h_\tau^{\text{LT}}\right) \qquad \left(0 \le \frac{\left|h_\tau^\theta\right|}{h_\tau^\sigma} \le 1\right) \tag{8}$$

This estimator describes a floating link ($h^\theta \approx 0$) by its long-term behavior and a massively changing link ($\left|h^\theta\right| \approx h^\sigma$) by its short-term estimate. Intermediate assessment is achieved using the relative behavior of trend and variation.

Approach number two is a confident long-term predictor:

$$H_\tau^{\text{pred}} = \begin{cases} h^{\text{LT}} + h^\theta - \omega \cdot h^\sigma, & \text{if } h^\theta \ge \omega \cdot h^\sigma \quad (0 \le \omega < 1) \\ h^{\text{LT}} + h^\theta + \omega \cdot h^\sigma, & \text{if } h^\theta \le -\omega \cdot h^\sigma \\ h^{\text{LT}} & \text{else} \end{cases} \tag{9}$$

In contrast to the previous method, a link is classified by the difference between trend and variation. If a clear trend can be identified—i.e., a relative threshold ω is exceeded—the long-term value is shifted correspondingly. The goal is to improve the prediction of the link behavior by incorporating the trend.

4 Evaluation Methodology

To evaluate and compare HoPS with the LQEs introduced in Sect. 2.2, we conducted simulations based on a 13-day real-world experiment. The purpose of this experiment is to gather real-world data in terms of the PSR and channel information, which is used by the TinyOS Simulator (TOSSIM) to feed the different LQEs. The main benefits of this approach are (i) detailed information about node connectivity in a real-world scenario, (ii) an identical runtime situation for all experiments, and (iii) repeatable experiments under real-world conditions.

4.1 Data Basis

A sensor-node testbed, consisting of 15 IRIS nodes with distances between 1 m and 40 m, was employed in our University building. All nodes were USB-powered. Every node broadcasted beacon messages within a fixed interval of 4 s with a random jitter using a uniform distribution. The transmit power was 0 dBm on radio channel 22. Packets consisted of a virtual payload of 13 bytes—containing a beacon sequence number—and a 13 byte MAC header. Clear Channel Assessment (CCA) was performed with at most 5 retries. We have not deactivated this feature, as we are interested in realistic connectivity behavior of a sensor network; moreover, a failed CCA implies that the sender experiences a high radio signal, but it does not necessarily imply the inability of the receiver to correctly receive a packet. Upon reception of a beacon, the following data is logged on the connected computer: node ID of sender and receiver, the sequence number, LQI, and SNR. From this data, a detailed track of packet receptions and corresponding physical channel quality was produced. More than 280 000 packets were transmitted per node.

4.2 Methodology and Metrics

We adjusted the physical layer of TOSSIM to reproduce our office environment. This layer utilizes the data basis to decide if a packet sent at a given time is received by other nodes in the network. When a node puts a radio packet onto the channel, this physical layer checks for all other nodes in the network, if they have received the experiment beacon in that time slot. The packet is only delivered to the nodes, for which this is the case. The result is a realistic, repeatable, and fair simulation environment.

We obtained the PSR ground truth by applying Hamming windows of lengths from 30 s to 60 min. The different LQEs are compared in terms of their ability to correctly represent the current and future course of each link. For this comparison, we used a normed cross correlation function (CCF) and the mean absolute error (MAE). Quality values are in the range of 0 to 100% with an 8-bit resolution; EETX values are converted to PSR. In case of F-LQE, the raw value is used. The implementations of LEEP and RNP have a resolution of tenth. ALE and HoPS run with 16-bit integers. To achieve fair conditions, we also ran an adapted version of LEEP using double precision.

4.3 Parameters

The parameters of the LQEs have been chosen as proposed in the corresponding papers. LEEP and RNP use an EWMA coefficient of 0.9 and update the estimations every 3 packets. F-LQE uses the fuzzy membership functions from [1] and a fuzzy parameter of 0.6. ALE uses filter coefficients of 0.9 in agile and 0.987 in stable link state, where PSR estimates of 86% and 74% serve as the thresholds to switch states. The parameters for HoPS are $\alpha = 0.9$ for short-term and $\beta = \gamma = 0.997$ for long-term and deviation estimation. HoPS initializes

$h^{\mathrm{ST}} = h^{\mathrm{LT}} = 50\%$ for new links. The threshold for the stable long-term prediction is $\omega = 0.25$. These parameters have been determined using a Java GUI that was developed for detailed link assessment and link-quality filter design.

5 Evaluation Results

In this section, the most important evaluation results are presented. First, the ability of different LQEs of link-quality tracking is analyzed and compared using the testbed data introduced in Sect. 4.1. Second, their performance in terms of correct link selection is evaluated.

5.1 Link-Quality Tracking

The estimation methods of the different LQEs vary largely, so that light has to be shed on their ability of link-quality tracking. In case of good links with a stable PSR above 95% and a SNR well above 5 dB, all LQEs achieve good estimates. If, in contrast, links have a dynamic behavior, this picture changes.

Medium Link. A medium link with a long-term PSR above 75%, notable short-term deviations, and SNR values below 5 dB is shown in Fig. 2. The second plot reveals that ALE tracks the long-term quality very well, but immediately toggles to the agile state, when its estimation falls below the 74% PSR threshold, and returns to the stable state after reaching the 86% PSR threshold. In the worst case, this may lead to oscillation between these states. F-LQE assigns a rather low value to the link, which is mainly caused by the low SNR values: the quality value is only about half the value of the long-term PSR. Instead of following the application-relevant PSR value, F-LQE mirrors the course of the SNR. This leads to the strong lag regarding the recovery phase towards the end of the displayed time window, where the short-term PSR is approaching the long-term measure even before the SNR value is increasing again. LEEP and RNP natively describe link-quality as EETX values with a resolution of tenth, so that their quality curve degenerates to a step function. RNP tracks variations of the link quality very quickly, but produces many overshoots. In contrast to this, LEEP offers a strong low-pass characteristic, but produces peak values. They occur on some rapid short-term PSR improvements. This behavior is alleviated, if the resolution of LEEP is increased (LEEP dbl), so that LEEP can track link quality more smoothly.

The short- and long-term estimations of HoPS follow the corresponding ground-truth trace accurately, giving a differentiated picture of the link's course. In addition, the variation value produced by HoPS resembles the absolute average deviation of the two estimates in the recent past. The latter corresponds to the variation of the ground truth in the top plot. Up to the middle of the time window, the HoPS trend correctly identifies the link to be floating. With a small time delay, the decreasing short-term link-quality is identified. Due to the rather small and slow change, trend and variation stay below an absolute value of 10%.

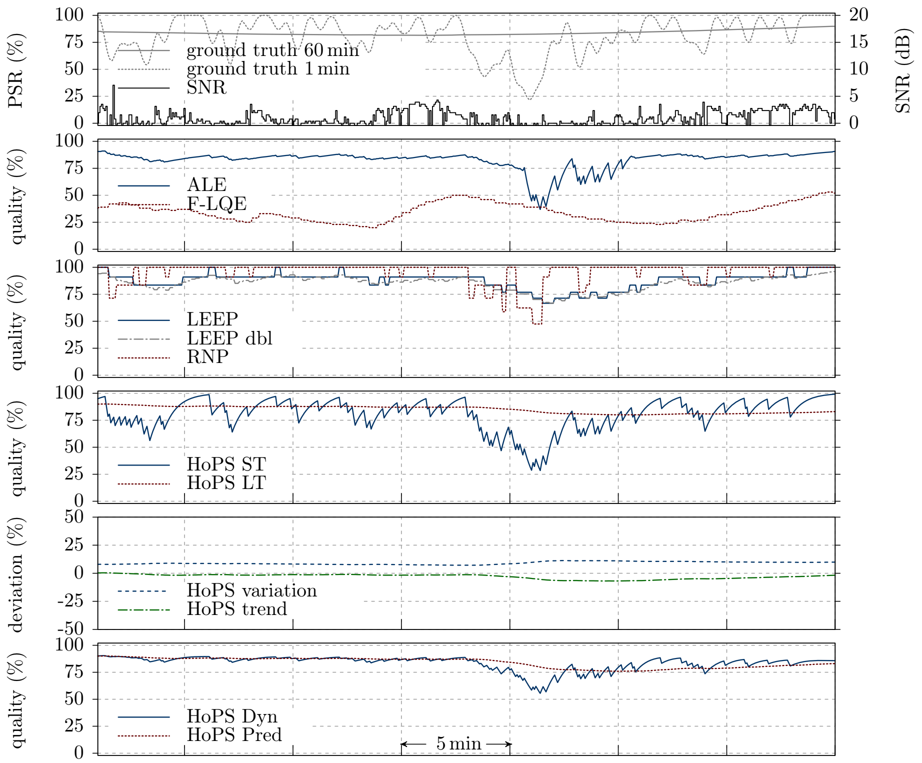


Fig. 2. LQE performance for a medium link with moderate to low SNR

After the short disturbance of the link, the trend approaches zero and the variation is decreasing towards its old value. HoPS Dyn is in a stable long-term state until the beginning of the disturbance. Due to the increasing absolute value of the trend at that point, HoPS Dyn takes an intermediate value between short- and long-term prediction of HoPS. During the link recovery phase, HoPS Dyn slowly returns to the stable prediction. In contrast, HoPS Pred is only marginally influenced by the disturbance, since it is too short and small.

Poor Link. A study of a deteriorating link reveals additional differences between the LQEs. The link displayed in Fig. 3 drops from a high, stable quality state to a completely useless link within less than 5 minutes.

ALE implicitly trusts the link for a few minutes due to the large filter coefficient. After falling below the 74% border line, ALE follows the quality drop quickly, but with a notable time-delay. The better a link has been, the larger this delay will be. Due to the adapted filter coefficient, the PSR bursts at the end of the displayed window are tracked closely. Although link tracking appears to work quite well, there are drawbacks. Firstly, there is no option to choose between short- and long-term estimation values. Secondly, random effects caused

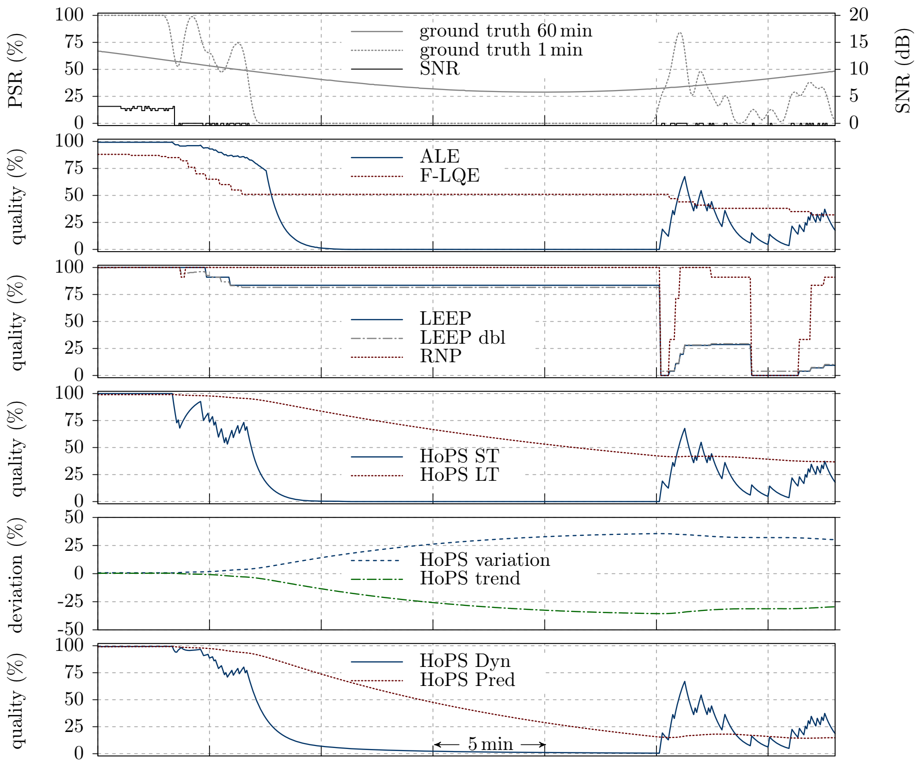
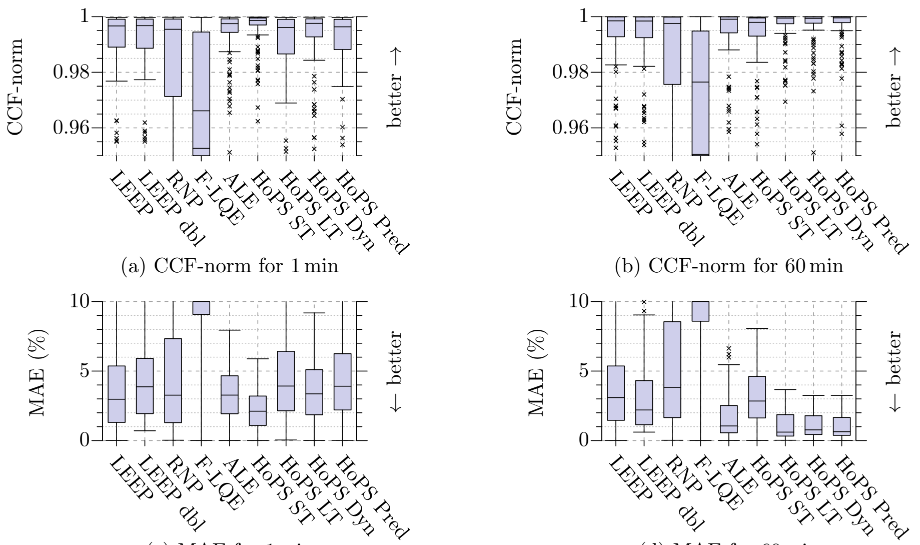


Fig. 3. LQE performance for a deteriorating link with low SNR

by the static thresholds for changing the link state may occur. The large peak after the long link disturbance misses the 86% threshold for changing to the stable state by less than 20%, which corresponds to the reception of only 2 packets.

F-LQE identifies the degenerating quality at the beginning accurately, since it coincides with a sudden SNR drop, but stops updating the quality estimate, because no more packets are received. Only upon the reception of new packets, the quality estimate is updated—which is happening quaintly when the link comes up again. Note that F-LQE assigns a larger link-quality value to the link in its broken state than to the medium link in the previous example. RNP and LEEP reveal the same weakness, plus RNP overestimates the link during its bursty phase at the end. For an application, the broken link would look like a perfect (RNP) or good (LEEP) link for more than 15 minutes.

In contrast, HoPS keeps track of the declining link-quality for both the short- and long-term values. Due to the large filter coefficient of HoPS LT, there is a notable lag. The question on which value to trust is answered by the variation and trend: They take large values that have about the same absolute value. This indicates a massive change of the link away from its long-term behavior. Due to the small slope of HoPS LT, trend and variation keep their large absolute values



(a) CCF-norm for 1 min

(b) CCF-norm for 60 min

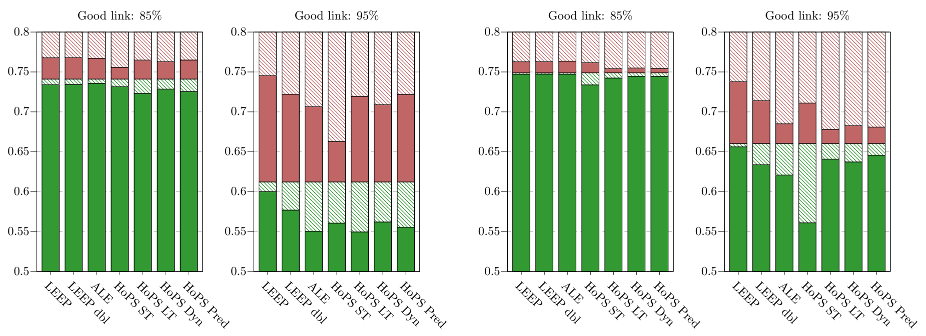
(c) MAE for 1 min

(d) MAE for 60 min

Fig. 4. Ability of the different LQEs to track the course of the empirical PSR

until the end of the disturbance. At this point they are decreasing again, since the link is revitalizing. HoPS Dyn detects the deterioration phase quickly and switches to the short-term estimate. The previous dip in link-quality is ignored due to the almost zero trend at this point. During the bursty phase at the end, HoPS Dyn sticks to the short-term estimate, thus reflecting the uncertain future of the link. Note that this behavior can be adjusted by the choice of the deviation filter coefficient γ. HoPS Pred reacts slowly to the broken state of the link, but has a similar course as the long-term ground truth in the first plot. However, it does not react to the improving link-quality in a suitable way.

Statistical Analysis. While these examples give a detailed understanding of the strengths and weaknesses of the LQEs, a statistical evaluation of all links is shown in Fig. 4. The box-and-whisker plots summarize the ability of link-quality tracking using the CCF-norm and the MAE (in absolute percent error) for all links in the testbed, on which at least one packet was received. For short-term estimation ALE, HoPS ST and HoPS Dyn give the best results, see Fig. 4(a) and 4(c). Here, HoPS ST gives the best results, because it always is in an agile mode, whereas ALE can only follow poor links quickly. Both RNP and F-LQE exhibit large variations in their ability to track link-quality. We are aware that F-LQE does not actually try to estimate the PSR, but from an application point of view, the PSR (or ETX) is the relevant link-quality metric. Because F-LQE has a very low correlation values, it cannot even track the relative behavior of the PSR, so that it is doubtful whether its values are useful for applications at all. A detailed view into the course of a few links reveals why LEEP is performing with a relatively low MAE: Many links have a quality-value close to 100% for long



(a) 1 min empirical PSR (b) 60 min empirical PSR

Fig. 5. Relative number of correct (solid) and wrong (hashed) link classifications (selections: bottom, rejections: top)

times. For these links, LEEP predicts a value of 0 EETX, yielding a 100% PSR and thus a low error. This partially cancels the heavy estimation errors when LEEP is not updating its quality metric. In contrast, pure EWMA approaches suffer from optimal links, since they are slow in taking extreme values.

Long-term link-quality estimation, see Fig. 4(b) and 4(d), is handled well by the same candidates as before. ALE and all HoPS estimators except HoPS ST give very low MAE values with a median of about 1%. 75% of the links are predicted correctly with an error less than 2%. This is a lower value than the best quarter of LEEP estimates. It comes as a little surprise that LEEP shows a comparable performance for both short- and long-term ground truth traces. It therefore turns out, that for long-term link-quality assessment a strong low-pass behavior is beneficial, because many links had only short disturbances. However, this picture may slightly change for a different testbed.

The outliers in the CCF-plots are caused by very weak links with short bursts of medium or high PSR values. We did not filter these links out manually, as they are realistic phenomena and a clear criterion for removement could not be found. It shows that HoPS performs a little better in the presence of these links.

5.2 Link Selection

In many application scenarios, nodes have to select links from a set of available candidates. This could be the case for selecting the next hop in a routing protocol or for deciding whether to keep or replace a possible neighbor in a size-limited neighborhood table. The latter example is of particular importance in dense networks, in which nodes can only track a small subset of all available communication partners due to memory restrictions and the lack of dynamic memory allocation in most sensor node operating systems.

Figure 5 shows the link-selection decisions of the LQEs for all available links and after each 4 s time interval for thresholds of 85% and 95% link-quality, respectively. If the current quality estimate of a link is equal to or above the threshold, that link is selected (solid areas), otherwise it is rejected (hashed areas). The decisions of the LQEs are compared to the real course of the link for 1 min and 60 min windows. Correct decisions are the solid areas at the bottom (correct selections) and the hashed areas at the top (correct rejections) of each bar. False rejections are represented by the lower hashed areas and false selections are indicated by the upper solid ones. The classification of the windows can also be read from the figures—e.g., Fig. 5(a) reveals that a fraction of 0.74 and 0.61, respectively, were above the two thresholds for the short-term window.

For the 1 min window in Fig. 5(a), HoPS ST achieves the smallest number of false decisions while HoPS LT generates the most errors due to its heavy smoothing. The other candidates produce comparable results for the 85% decision threshold. For the 95% one, LEEP makes a large number of false selections, because its resolution in this area is too low. Using a finer (LEEP dbl) resolution, this number is decreased at the cost of an increased amount of false rejections. In terms of cluster formation and long-term routing decisions, false selections may pose a severe hazard on throughput and network stability, whereas false rejections lead to a negative impact only in sparse networks with very few links. Here, network connectivity can be prevented, if there are too many false rejections. HoPS ST and Dyn achieve better selection results for the 95% decision threshold than ALE, where HoPS Dyn produces more false selections than HoPS ST.

The 60 min window in Fig. 5(b) flips the coin in favor of the long-term HoPS solutions HoPS LT and HoPS Pred. For the decision threshold of 85% they leave out some good links, but produce way less false selections than LEEP and ALE, which have the same performance in this setting. The 95% threshold gives an even more differentiated picture. HoPS Pred and HoPS LT make few wrong selections and rejections. HoPS Dyn produces comparable results; while falsely selecting almost as many links as ALE, the number of false rejections is only half as large. HoPS ST is too agile to produce good results. Increasing the resolution of LEEP decreases the number of false selections insufficiently.

To investigate the decisions of the LQEs in more detail, Fig. 6 portrays the relative distribution of selected (solid bars) and rejected (hashed bars) links for the long-term ground truth with 95% decision threshold. The solid curve indicates the Cumulative Distribution Function (CDF) for selected links, i.e., the fraction of selected links with an actual PSR on the X-coordinate. Similarly, the dash-dotted curve indicates the inverse CDF for rejected links, i.e., the fraction of rejected links with an actual PSR greater than the value on the X-coordinate. The crossing points at the 95% vertical borderline resemble the decision accuracy.

10% of the links selected by LEEP have an actual PSR less than 95%, while almost no links with a PSR of at least 97% have been rejected. The opposite case is true for ALE. More than 10% of the rejected links are actually better than 95%. At the same time, link selection works well: Less than 4% of the falsely selected links turn out to have a PSR below 94%. The selection CDF for HoPS Dyn and HoPS Pred follows almost the same course as the one of ALE, but the rejection CDF is lower—approximately 7% for HoPS Pred and 5% for HoPS Dyn. The two HoPS derivatives therefore exhibit a better rejection behavior of links below the

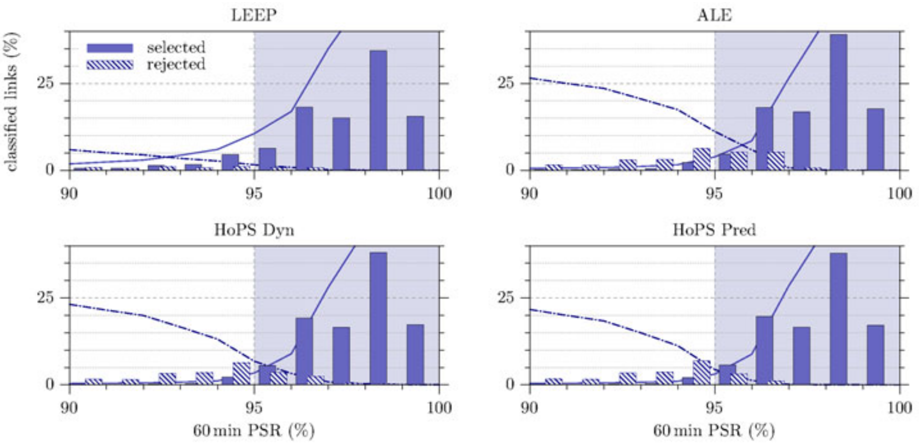


Fig. 6. Distribution of 60 min link PSRs for selected (solid) and rejected (hashed) links based on a decision threshold of 95% PSR (good links)

decision threshold by incorporating the trend and variation. The main difference between HoPS Dyn and HoPS Pred is that the latter has identified the links in the region between 96% and 97% more accurately.

6 Conclusion

In this paper, we have presented a novel approach on link-quality estimation in wireless sensor networks: the Holistic Packet Statistics (HoPS). This new LQE calculates four distinct quality metrics, describing the short- and long-term quality of a link at the same time, while also providing information about the dynamics of a link by means of the variation and trend of the link-quality. All of these metrics can be accessed by the application. These metrics are calculated efficiently using EWMA filters and hence have a small memory footprint not exceeding that of well-known competitors, such as F-LQE or LEEP.

HoPS has been compared to a bouquet of existing LQEs by using PSR data gathered in an indoor testbed. Due to its four metrics, HoPS tracks the progress of a link accurately, being therefore able to inform an application about the course of short- and long-term link-quality. An adaptive link-quality estimate can be provided by taking the variation and trend into consideration. This is supported by a comparative study of PSR traces of various links. An additional analysis of the correlation and average absolute error of real PSR values proves the gain of HoPS over existing LQEs. Moreover, it has been shown that HoPS improves link selection for routing decisions or neighborhood formation.

However, open issues and future work have been identified. Firstly, useful link-quality prediction is only possible, if link behavior is studied in more detail—e.g., sensing that a good link is currently deteriorating is more useful, if it is known in how many cases a link recovers quickly versus the times it breaks down completely. Secondly, the potential of HoPS can only be fully exploited, if its metrics are actually used by the application. Merging the four values into a single prediction value brings low improvements only. Therefore, advanced routing and neighborhood management protocols have to be devised. We plan to make further investigations on this ground, hoping to push link-quality assessment in wireless sensor networks a step forward.

Acknowledgments. We thank the anonymous reviewers for their insightful feedback. This material is based upon work partially supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) under contract number TU 221/4-1.

References

1. Baccour, N., Koubâa, A., Youssef, H., Ben Jamâa, M., do Rosário, D., Alves, M., Becker, L.B.: F-LQE: A Fuzzy Link Quality Estimator for Wireless Sensor Networks. In: Silva, J.S., Krishnamachari, B., Boavida, F. (eds.) EWSN 2010. LNCS, vol. 5970, pp. 240–255. Springer, Heidelberg (2010)
2. Becher, A., Landsiedel, O., Kunz, G., Wehrle, K.: Towards Short-Term Wireless Link Quality Estimation. In: Hot EmNets (March 2008)
3. Caleffi, M., Paura, L.: Bio-inspired Link Quality Estimation for Wireless Mesh Networks. In: WoWMoM (June 2009)
4. 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 (May 2005)
5. Chen, Y., Terzis, A.: On the Mechanisms and Effects of Calibrating RSSI Measurements for 802.15.4 Radios. In: Silva, J.S., Krishnamachari, B., Boavida, F. (eds.) EWSN 2010. LNCS, vol. 5970, pp. 256–271. Springer, Heidelberg (2010)
6. Couto, D.S.J.D., Aguayo, D., Bicket, J., Morris, R.: A High-Throughput Path Metric for Multi-Hop Wireless Routing. In: MobiCom (September 2003)
7. Fonseca, R., Gnawali, O., Jamieson, K., Levis, P.: Four-Bit Wireless Link Estimation. In: HotNets VI (November 2007)
8. Gnawali, O.: The Link Estimation Exchange Protocol (LEEP) (February 2006), `http://www.tinyos.net/tinyos-2.x/doc/html/tep124.html`
9. 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: GlobeCom (December 2003)
10. Liu, T., Kamthe, A., Jiang, L., Cerpa, A.: Performance Evaluation of Link Quality Estimation Metrics for Static Multihop Wireless Sensor Networks. In: SECON (June 2009)
11. Senel, M., Chintalapudi, K., Lal, D., Keshavarzian, A., Coyle, E.J.: A Kalman Filter Based Link Quality Estimation Scheme for Wireless Sensor Networks. In: GlobeCom (November 2007)
12. Srinivasan, K., Levis, P.: RSSI is Under Appreciated. In: EmNets 2006 (May 2006)
13. Weyer, C., Unterschütz, S., Turau, V.: Connectivity-aware Neighborhood Management Protocol in Wireless Sensor Networks. In: FGSN (September 2008)
14. Woo, A., Tong, T., Culler, D.: Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In: SenSys (November 2003)

The Impact of Network Topology on Collection Performance

Daniele Puccinelli[1,⋆], Omprakash Gnawali[2,⋆], SunHee Yoon[2], Silvia Santini[3], Ugo Colesanti[4], Silvia Giordano[1], and Leonidas Guibas[2]

[1] University of Applied Sciences of Southern Switzerland, Switzerland
{daniele.puccinelli,silvia.giordano}@supsi.ch
[2] Stanford University, USA
{gnawali,guibas}@cs.stanford.edu, sunhee@stanford.edu
[3] ETH Zurich, Switzerland
santinis@inf.ethz.ch
[4] Sapienza University of Rome, Italy
colesanti@dis.uniroma1.it

Abstract. The network topology has a significant impact on the performance of collection protocols in wireless sensor networks. In this paper, we introduce an unobtrusive methodology to quantify the impact of the topology on the performance of collection protocols. Specifically, we propose a protocol-independent metric, the Expected Network Delivery, that quantifies the delivery performance that a collection protocol can be expected to achieve given the network topology. Experimental evidence obtained with two collection protocols on numerous topologies on testbeds shows that our approach enables a systematic evaluation of protocol performance.

1 Introduction

The rich and active research in network protocols in Wireless Sensor Networks (WSNs) has progressively emphasized the testbed evaluation of protocols over simulation. Several testbeds exist with a hundred or more mote-class nodes. The use of these testbeds has led to protocols that can function in the harsh environment they often encounter in real-world deployments. The Collection Tree Protocol (CTP) [7], for example, was adopted in several deployments [2][9] due to the promising results it achieved on a large number of testbeds.

Experiments on a testbed subject a network protocol to the vagaries of real-world wireless links [17], with no approximations or simplifying assumptions about their behavior. This is a clear improvement over simulation. The uncertainty in the behavior of wireless links is valuable for protocol evaluation, but it also represents a drawback of testbed experiments compared to simulations. Network simulations offer a fine-grained control of the network environment and the propagation conditions. In contrast, testbed users can only test their protocols on specific, non-reproducible situations, because they have almost no control over the state of the network.

⋆ Lead Authors.

In a wireless network, the topology is jointly determined by the *network layout* and the *link dynamics*. The effective topology over which routing paths are established also depends on the choice of routing destination, which corresponds to the *sink placement* in the context of WSNs. The combination of the network layout, the link dynamics, and the sink placement, which we simply refer to as *network topology*, has a large impact on protocol performance.

The performance of a protocol is a function of the topology as well as of the protocol's own mechanisms. Thus, we cannot attribute the performance achieved by a protocol entirely to its mechanisms without considering the state of the network. This makes it challenging to reason about protocol performance on a testbed. Figure 1 shows that both CTP and the Arbutus collection protocol [12] achieve a wide range of delivery ratio and goodput levels even on a single testbed. With both protocols, we observe a dichotomy between high-performing and low-performing topologies, which we refer to, respectively, as Class A and Class B topologies. Due to the lack of a methodology to describe the topology on which a testbed experiment is performed, even in papers where protocols are compared experimentally on real-world testbeds, there is at most a quick comment on the topology used. Different experiments may have been run over different network topologies, which makes it difficult for the community to reproduce the testbed results or to systematically reason about the differences in protocol performance across testbeds.

To cope with the lack of control over the state of a testbed across multiple experiments, we propose to explicitly capture the state of the network while evaluating protocols on the testbeds. For this purpose, we introduce a protocol-independent network metric, the Expected Network Delivery (END), that captures the reliability of the achievable routing paths from each node to the sink. The END quantifies the delivery performance that a collection protocol can be expected to achieve given the network topology. This metric helps decouple the impact of the network topology from the impact of the protocol's own mechanisms on collection routing performance.

Using the END to characterize the network enables a systematic testbed evaluation of network protocols despite the lack of control over the testbed topology. A collection protocol, for example, might achieve different delivery ratios when tested on different testbeds or even on the same testbed at different times. If the END changes significantly across multiple experiments, changes in the network topology can explain the performance variations. On the other hand, if the END remains stable, the difference in performance can be attributed to the mechanisms in the protocol that reacted differently on different experiments despite the network state being roughly the same. Moreover, the range of END values across different experiments captures the range of network conditions encountered during the experiments. If we test a protocol across a large number of testbeds but only span a narrow END range, then we have failed to test the protocol across a wide range of network conditions.

We have run a large number of experiments with two different collection protocols, CTP [7] and Arbutus [12], on the Motelab [23] testbed over a period

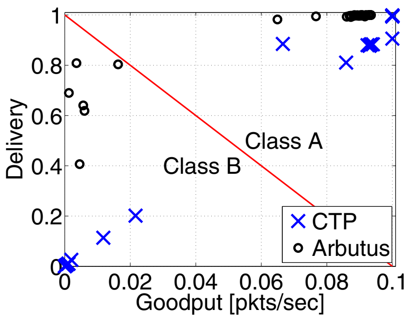


Fig. 1. The performance of collection protocols in MoteLab varies significantly depending on the sink placement, as shown by these results obtained with CTP and Arbutus. There are two distinct performance classes, which we label as A and B.

of several months. Furthermore, we have tested the performance of CTP on the Castalia wireless sensor network simulator [1]. We observed that different combinations of protocol, sink placement, testbed, and experiment time results in a wide range of performance. With the END computed during these experiments, we were able to conclude that the performance variations were primarily due to the properties of the topology present during those experiments rather than the protocol mechanisms.

In this paper, we make these contributions:

- We show that the performance of collection protocols on testbeds depends on the network topology at the experiment time.
- We design the END, a protocol-independent metric to capture the key properties of the network topology that affect the performance of the protocol.
- We propose a methodology to systematically evaluate the performance of a protocol across various testbeds, topologies, and experiments despite having no control over the network dynamics on the testbeds.
- We evaluate the effectiveness of the END, by analyzing the results from a large number of testbed experiments as well as simulations, with CTP and Arbutus collection protocols as examples.
- We show that our methodology is applicable not only to collection, but also to other categories of protocols.

2 Quantifying the Impact of the Topology

In this section we define the Expected Network Delivery, our primary topology-aware metric, along with a secondary metric called Balanced Delivery. We illustrate that these metrics quantify the impact of the network topology on collection performance by capturing the impact of the key links in the network given the node layout and the sink placement.

2.1 Expected Network Delivery

Let $\mathcal{N} \subset \mathbb{N}$ denote the set of nodes in a WSN. We assume a many-to-one traffic flow to a sink $s \in \mathcal{N}$ enforced by an arbitrary distributed routing protocol. When node i transmits to node j, they form a directional wireless link that we denote as (i, j). We use a comma-separated list of nodes within square brackets to denote a route; for instance, if i uses j as a relay to get its packets to s, the corresponding two-hop route is represented as $[i, j, s]$. We define the (one hop) link Packet Reception Ratio (PRR) over the link (i, j), $\pi_{i,j}$, as the fraction of the packets transmitted by i that were directly received by j (*i.e.*, over one hop) over a given time window T. The link PRR values collectively give us a snapshot of the network connectivity over T. We account for asymmetric links by using $\lambda_{i,j} \triangleq \min(\pi_{i,j}, \pi_{j,i})$ as the PRR for the link (i, j). We refrain from using an ETX-like metric such as the product $\pi_{i,j}\pi_{j,i}$ because the forward and the reverse channel are not independent [3].

Given the specific sink placement, each node employs a distributed routing protocol to find a route to the sink. We assume that the protocol's goal is to maximize the delivery of data packets to the sink. To capture the state of the network, we acquire network connectivity data while the protocol is running and, after the completion of the experiment, compute the link PRRs and apply Dijkstra's algorithm [5] with $1/\lambda_{i,j}$ as the link metric to obtain the paths from each node to the sink that maximize the overall delivery to the sink. We then compute the Expected Path Delivery (EPD) e_k from node k to the sink s as

$$e_k = \Pi_{h=0}^{H-1} \lambda_{r_h, r_{h+1}}, \tag{1}$$

where r_h represents the h^{th} hop between k and s (with $r_0 \triangleq k$ and $r_H \triangleq s$), and H denotes the number of links that form the route between k and s. In order to quantify the expected performance of a collection protocol with a global knowledge of the network topology, we define our topology-aware collection metric, the **Expected Network Delivery** (**END**), denoted as END $\in [0, 1]$, as the expected path delivery averaged over all nodes:

$$\text{END} = \frac{1}{|N|} \sum_{k \in \mathcal{N}} e_k. \tag{2}$$

The END is therefore a function of the link PRRs, which capture the net effect of all the vagaries of wireless propagation. For this reason, our metric captures the ground truth of the state of the network and describes the network topology in a protocol-independent fashion. The END captures the impact of the link connectivity on a network-wide level, distinguishing the key links from the redundant ones. Though in this paper we focus on many-to-one traffic, our methodology is based on the connectivity properties of the network and could be applied to any traffic pattern.

Although our metric is protocol-independent, it is necessary to extract it while a given protocol is running so that we can capture the properties of transitional links during the experiment. If a topology is dominated by unstable links whose coherence time is lower than or comparable to the duration of the experiment, then even capturing connectivity data right before and right after the experiment would be misleading. Since WSN routing protocols typically employ broadcast control traffic for topology discovery and route maintenance,we obtain our metrics by computing the PRR measurements based on the protocol's control traffic, which is acquired over the testbed's backchannel. This approach is non-intrusive, because it relies on passive measurements that do not interfere with the protocol. We ignore all protocol-specific information, such as, for instance, the contents of the neighbor tables.

Since the END is computed by using $\lambda_{i,j} \triangleq \min(\pi_{i,j}, \pi_{j,i})$ as the PRR for the link (i, j), the END is insensitive to the direction of the network traffic. For this reason, we complement the END with a secondary metric, the **Balanced Delivery (BD)**. The BD, denoted as $B_s \in [-1, 1]$, is defined as:

$$B_s = E_s^{(\mathrm{out})} - E_s^{(\mathrm{in})}, \tag{3}$$

where $E_s^{(\mathrm{out})} \in [0, 1]$ is the *Outbound* Expected Network Delivery (O-END) of the sink s, and $E_s^{(\mathrm{in})} \in [0, 1]$ is the *Inbound* Expected Network Delivery (I-END) of the sink s. The I-END is obtained by applying Dijkstra's algorithm with $\lambda_{i,j} \triangleq \pi_{i,j}$, while the O-END is obtained by applying Dijkstra's algorithm with $\lambda_{i,j} \triangleq \pi_{j,i}$.

2.2 Capturing the Impact of the Key Links

We use the network shown in Fig. 2 as an example to explain how the proposed metrics are computed. In the figure, the value on each directional link indicates the PRR. In Table 1, we report the optimal route from each node k to the sink s obtained with Dijkstra's algorithm, along with the corresponding expected path delivery e_k with respect to the appropriate link metric. The END, I-END, and O-END are obtained by averaging out the expected path deliveries over all nodes.

The distribution of the link PRR for all the links in the network might seem like a promising alternative to the END. A network with a large number of high quality links should result in a better protocol performance. However, the protocol performance depends on the quality of the links that the protocol uses and not on the quality of the remaining links. We use the expression *key links* to indicate those links whose absence would partition the network or force the routing protocol to use unreliable links. Because efficient and reliable routing protocols select key links, the END is designed to capture their impact. To clarify this point, let us perturb the network in Fig. 2 in different ways to see how the END responds as opposed to the mean link PRR.

1. *An unreliable key link becomes reliable.* The improvement of a key link is a huge benefit to the network, and so the value of a valid topology-aware metric should increase significantly. Link (s, 2) is a key link with a low PRR. If the PRR of this link increases to 1, the END increases by more than 66% (from 0.4 to 0.67), while the mean link PRR only changes slightly (from 0.67 to 0.69).

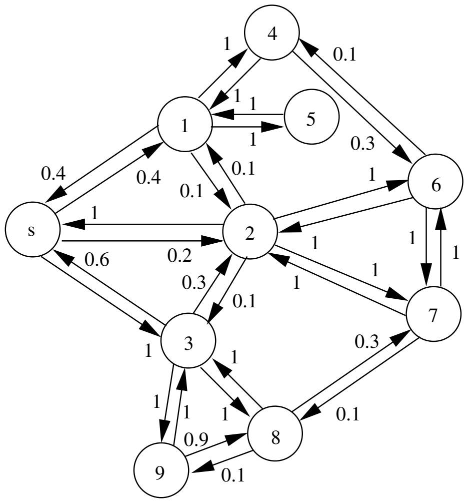


Fig. 2. A sample network with challenging connectivity conditions. The numeric values next to the arrows represent the PRR of the corresponding links in the direction of the arrow.

Table 1. Expected path delivery (EPD) values for the nodes in the network in Fig. 2. The EPD is computed in three different ways: with $\lambda_{i,j} \triangleq \min(\pi_{i,j}, \pi_{j,i})$ to obtain the END, with $\lambda_{i,j} \triangleq \pi_{i,j}$ to obtain the I-END, and with $\lambda_{i,j} \triangleq \pi_{j,i}$ to obtain the O-END. The END, I-END, and O-END are computed as the average of the corresponding EPDs.

Node k	**Route**	e_k		
		$\lambda_{i,j} \triangleq \min(\pi_{i,j}, \pi_{j,i})$	$\lambda_{i,j} \triangleq \pi_{i,j}$	$\lambda_{i,j} \triangleq \pi_{j,i}$
1	[1, s]	0.4	0.4	0.4
2	[2, s]	0.2	1	0.2
3	[3, s]	0.6	0.6	1
4	[4, 1, s]	0.4	0.4	0.4
5	[5, 1, s]	0.4	0.4	0.4
6	[6, 2, s]	0.2	1	0.2
7	[7, 2, s]	0.2	1	0.2
8	[8, 3, s]	0.6	0.6	1
9	[9, 3, s]	0.6	0.6	1
		END=0.4	I-END=0.67	O-END=0.57

2. *A reliable key link becomes unreliable.* Link (3, s) is the most reliable key link in the network, although its PRR is just 0.6. If we set $\pi_{3,s} = 0$, the END drops 50% (from 0.4 to 0.2), while the mean link PRR remains virtually unchanged. The drop in the END in response to the worsening of a key link is proportional to the relative importance of the link. For instance, link (3, 8) is only used for route [8, 3, s] and is not as critical as (3, s); if we set $\pi_{3,8} = 0$, the END only decreases by 15% (from 0.4 to 0.34).

These examples illustrate that the added value of the END comes from its ability to distinguish the links that matter from those that do not.

3 Network Topology and Protocol Performance

In this section, we show how the END and BD metrics make it possible to isolate and better understand the impact of the network topology on protocol performance. We also explore the generality of these metrics by applying them to simulation studies of collection and point-to-point routing protocols.

3.1 Experiments and Metrics

Table 2 shows an overview of the experiment sets used in this paper (the results in Fig. 1 are from the experiments in motelab-arbutus and motelab-ctp). We also performed a smaller number of experiments on the Tutornet testbed, as well as simulations on TOSSIM [10] and Castalia [1].

Table 2. Overview of the experiment sets used in the paper

Experiment set	**Testbed**	**Routing**	**IPI [sec]**	**IBI [min]**	**Points**	**Duration [hrs]**			
						ave	min	max	tot
motelab-ctp	MoteLab	CTP	10	Trickle	18	1	1	1	18
motelab-arbutus	MoteLab	Arbutus	10	1	32	0.5	0.2	1	16

In our experiments, each node injects packets at a constant Inter-Packet Interval (IPI) value towards the single destination, the sink. The IPI includes a small jitter to avoid packet synchronization across the nodes. The routing protocols broadcast their own control messages, known as beacons, at a given Inter-Beacon Interval (IBI), which is fixed for Arbutus and variable for CTP, which employs adaptive beaconing [7]. In this study, we use the performance metrics typically employed in evaluation of routing protocols:

– Delivery Ratio: The ratio of the number of packets that are delivered to the sink to the total number of injected packets.
– Goodput: Number of application packets delivered to the sink per node per unit time (here measured in pkts/sec).
– Delay: The time it takes for a packet to travel from the source to its destination.
– Cost: The total number of transmissions (including retransmissions) needed to get a packet from its source to the sink.

During each experiment, we log all control beacons and use them offline to measure the PRR for all the network links. In turn, the measured PRR is employed to compute the END metric for the experiment at hand.

Table 3. END and delivery ratio for four examples of MoteLab topologies

Sink	Class	END	Delivery Ratio
46	A	0.73	0.9984
25	A	0.38	0.9864
22	B	0.18	0.8722
90	B	0.05	0.5119

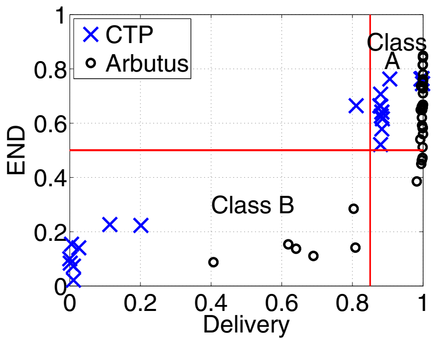


Fig. 3. With both CTP and Arbutus, the END metric generally correlates with the delivery rate, and the performance dichotomy shown in Fig. 1 is confirmed. Low END values correspond to a less predictable performance.

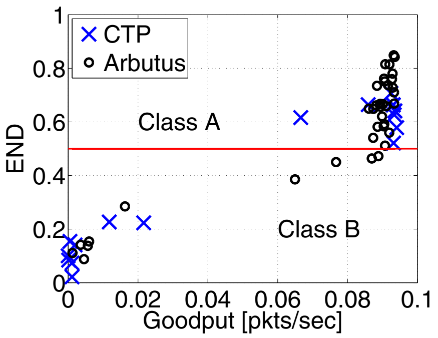


Fig. 4. For both CTP and Arbutus, the END predicts the performance dichotomy between Class A and Class B also in goodput. In general, the better the END, the higher the goodput.

3.2 Protocol Performance and Topology

Table 3 shows the values of the END metric and the delivery ratio from four examples of MoteLab topologies from motelab-arbutus that yield very different performance levels. The delivery ratios range from 99.84% with sink 46 to about 51% with sink 90. Across the experiments, there is a distinct correlation between higher delivery ratio and higher END.

Figure 3 shows the delivery ratio vs. the END metric for the experiments in motelab-ctp and motelab-arbutus. Qualitatively, we observe a correlation between the END metric and the delivery ratio for both CTP and Arbutus; this confirms that the performance variations across different experiments may be traced back to changes in the network topology. With a lower END, the best possible achievable performance is also lower, as reflected in the protocol performance.

Table 4 and Table 5 summarize the results from the experiments in motelab-ctp and motelab-arbutus. We note that the dichotomy between Class A and Class B that was evident in Fig. 1 is also visible in these results. We can use the END metric to classify the topologies according to the expected achievable

Table 4. Results from the experiment set motelab-ctp. Performance of CTP at IPI=10s averaged over different END ranges: Class B, Class A, and its subclasses, A+ (upper Class A) and A- (lower Class A).

END Range	**Delivery Ratio**		**Goodput [pkts/sec]**	**Path Length**		**Cost**	
	mean	σ	mean	mean	σ	mean	σ
[0, 0.5) (B)	0.06	0.09	0.61e-3	7.1	2.5	23.7	18.9
[0.5, 0.9) (A-)	0.9246	1.47e-2	9.2e-2	4.1	0.9	5.5	2.8
[0.9, 1] (A+)	0.9997	2.33e-4	9.9e-2	3.2	0.2	3.7	0.2
[0.5, 1] (A)	0.9361	0.31e-2	9.3e-2	4.0	0.9	5.2	0.9

Table 5. Results from the experiment set motelab-arbutus. Performance of Arbutus at IPI=10s averaged over different END ranges.

END Range	**Delivery Ratio**		**Goodput [pkt/sec]**	**Path Length**		**Cost**		**Delay [sec]**	
	mean	σ	mean	mean	σ	mean	σ	mean	σ
[0, 0.5) (B)	0.66	0.15	0.6e-2	5.0	1.5	10.9	3.4	241.7	240.4
[0.5, 0.9) (A-)	0.9967	4e-3	8.8e-2	2.9	0.5	8.9	5.5	1.3	1.5
[0.9, 1] (A+)	0.9996	2e-4	9.3e-2	2.9	0.5	3.3	0.6	0.2	0.1
[0.5, 1] (A)	0.9975	3.6e-3	8.9e-2	2.9	0.5	7.3	5.3	1.0	1.3

performance. The END also correlates to various degrees with other metrics such as goodput, cost, and delay.

The END metric helps us understand the protocol performance in the context of the network topology over which an experiment is run. With the END metric, we can precisely identify the cases where a low protocol performance is due to an adverse network topology.

3.3 Explaining Protocol Performance

The END metric can help explain the reasons behind the achieved protocol performance. For example, in Class B topologies Arbutus performs more efficiently than CTP due to the use of different retransmission strategies: Arbutus employs unconstrained retransmissions (compared to 32 times for CTP) and limits packet loss at the price of delay, as shown in Table 5. In Class A topologies, CTP and Arbutus perform more similarly and achieve high delivery ratio. We expect this result considering the abundance of high quality key links in Class A topologies.

Figure 5 shows the packet loss and the corresponding END with a specific sink placement (node 22) in MoteLab. Each datapoint represents the packet loss and the value of the END metric over one of 50 experiments. There is a clear negative correlation between the END and the packet loss.

In Class B topologies, a high delivery ratio comes with a high cost because a large number of retransmissions is needed to deliver packets over unstable key links. In Class A, however, a high delivery ratio does not imply a high cost because the key links in Class A topologies have a high PRR and generally require a single transmission.

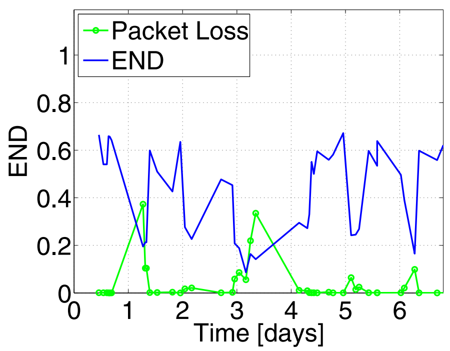


Fig. 5. An unstable sink assignment (MoteLab's 22) results in different network topologies that yield significant performance variations

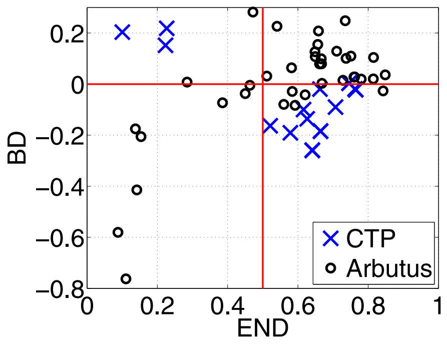


Fig. 6. Experiment sets motelab-ctp and motelab-arbutus: Balanced Delivery vs. END

3.4 Comparisons across Testbeds

The END metric enables a direct comparison of results obtained from different testbeds thereby overcoming the biggest shortcomings in testbed experimentation – the inability to directly compare the results from different testbeds. The results obtained on different testbeds are directly comparable if the END metrics across those experiments are similar.

In one experiment run on the Tutornet testbed, CTP achieved a delivery of 99.9% with an END of 0.89. Both CTP and Arbutus performed similarly in the MoteLab runs from motelab-ctp and motelab-arbutus when the END was in that same ballpark. Because the END values across these experiments on two different testbeds are similar, we know that the network topologies during these experiments were similar, and these two performance results are directly comparable. Thus, the END metric tells us when the topologies on two testbeds are similar and gives us a way to directly compare the results from two testbeds.

3.5 Comparisons over Time

Even if the sink placement and the network layout are fixed, protocol performance can still change over time due to the temporal changes in the link qualities in the network. Figure 5 shows that a testbed can have a time-varying topology that yields a time-varying performance. The performance peaks correspond to END maxima, while the performance lows map to END minima. We conjecture that this is due to the high impact of transitional links with this particular sink assignment: transitional links are more likely to get stuck in bad fading spots at night than they are during the day, when they can leverage induced fading effects [11].

This example shows that the END can also be employed for a systematic evaluation of a single protocol over time.

3.6 Directionality and Outliers

The quality of the links to the sink's neighbors, to a large extent, determines the performance of a collection protocol. A link can have bidirectional loss, dominantly outbound loss, and dominantly inbound loss. The nodes cannot send data to the sink if the links from these neighbors of the sink have high bi-directional or inbound losses, while acknowledgments and control packets from the sink tend to get dropped with high outbound loss.

Figure 6 shows the BD vs. the END for each of the motelab-ctp and motelab-arbutus experiments. We observe that the outbound losses are common in Class B topologies. Both protocols suffer significant outbound losses in those topologies. Because it contains several mechanisms to boost reliability, Arbutus performs significantly better than CTP with Class B topologies. Figure 6 shows that a near-zero BD always corresponds to a high END and therefore to high delivery ratios. We also found that near-zero BD are rare, suggesting that most links were unstable and asymmetric during our experiments. Strong dominantly inbound losses were never observed in our experiments. The corresponding topologies would result in near-zero packet delivery.

Moderate inbound loss (BD> 0) typically indicates the presence of connectivity outliers. If the END is very high (typically > 0.8), a positive BD is indicative of the presence of leaf connectivity outliers, *i.e.*, nodes with poor downstream links that are attached to the collection tree as leaf nodes. These leaf outliers result in a positive BD in Fig. 6. The positive BD allows us to determine that the dominant cause of CTP's poor performance is downstream loss. Thus, the BD metric enhances the performance analysis by adding directionality to the overall topology information captured by END.

3.7 Applicability to Simulation

Even an accurate qualitative description of a simulation setup makes it difficult to quickly determine the impact of the simulation environment on the protocol performance. Instead, the END metric can be used to succinctly capture the topology information used in simulations. Figure 7 shows the values of the END and the delivery ratio obtained by running CTP on 50 different network topologies (each consisting of 100 nodes) within the Castalia simulation environment [21][1]. For each network, the END and the delivery ratio were averaged over 50 simulation runs. Similarly to the testbed experiments, the END metric and the delivery ratio from the simulations show a significant degree of correlation. In this specific case, low END values correspond to relatively high average delivery ratios because in the simulations the channel remains constant across retransmissions. Because the END metric succinctly captures the property of the topology instantiated during the simulation, it allows us to understand the impact of the topology on the protocol performance in simulation.

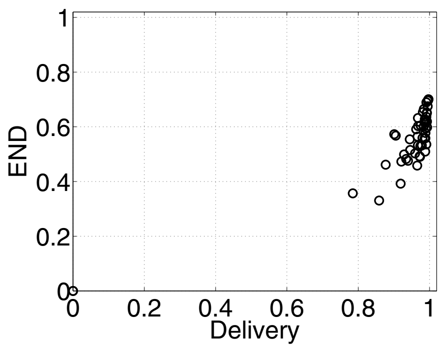


Fig. 7. END vs. delivery in a series of Castalia simulation runs of CTP. Each circle represents the average over 50 simulation runs with a given 100-node network.

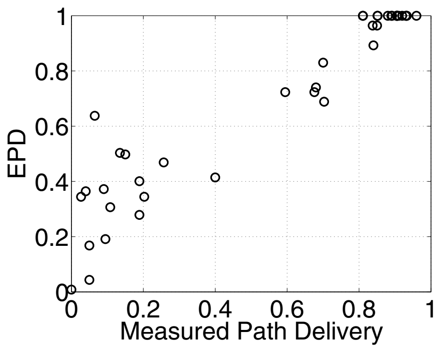


Fig. 8. Expected Path Delivery vs. measured path delivery in a series of TOSSIM simulation runs of TYMO, a point-to-point routing protocol (each circle represents one simulation run)

3.8 Applicability Beyond Collection

The definition of the END given in equation (2) presupposes a many-to-one traffic pattern. This formulation is specific to collection, but the framework is more generally applicable. For example, in the case of point-to-point routing, the Expected Path Delivery (EPD) given in equation (1) can be employed to gauge the expected performance on a route between two nodes. Figure 8 shows the results of a TOSSIM simulation of TYMO, a TinyOS implementation of the Dynamic MANET On-demand (DYMO) routing protocol [4]. Each circle represents one simulation run, and each run has a different set of link dynamics. These simulation results show that the EPD correlates well with the measured path delivery, and two performance classes can be identified as was the case with the testbed experiments in Fig. 3.

3.9 Limitations

Though the END and the BD are protocol-independent in their definition, their calculation leverages the protocol's control traffic. There is arguably some residual dependence on the protocol, mainly because we need to leverage the protocol's traffic to measure connectivity. Measuring the accuracy of the computed PRR would require the injection of additional traffic, which would affect the protocol's performance and perturb the results. This is a fundamental limitation of our framework that is due to the need to measure the network as we use it [8]. Another limitation lies in the fact that the END is averaged over the duration of each experiment. In networks where bimodal links dominate, or in long experiments, a time-dependent formulation of the END is in order and will be addressed in our future work.

4 Related Work

The impact of the network topology on protocols has been studied in the context of wired networks, with a specific focus on the Internet. Early studies considered the node degree distribution and the neighborhood size [6]. In [13], network topology is characterized with three metrics: the *expansion* (average number of nodes within a given hop count), the *resilience* (minimum cut-set size for a balanced bipartition of the network), and the *distortion* (which captures how path lengths are affected by link failures). Our study, however, is specific to wireless sensor networks, whose low-power communication hardware underscores the probabilistic nature of wireless links [17] and makes it impossible to treat them as Boolean objects (as in wired networks).

Recently, the lack of a *wireless lexicon* to describe the complexities of real-world wireless networks has been pointed out, and there have been a few efforts on the definition of link-level parameters that capture the vagaries of the behavior of low-end wireless network. In [19], a measure of link bimodality (the β factor) is defined, and its impact on protocol performance is characterized. In [18], a measure of inter-link cross-correlation (the κ factor) is proposed. A related effort is the development of the Stanford Wireless Analysis Tool (SWAT) [20], a software tool for the collection of network measurements. In these studies, network measurements are taken by injecting special traffic patterns: broadcast traffic in [19] and a round-robin of packet bursts in [18]). Our effort can be viewed as complementary to these studies, because (1) we focus on a network-wide metric as opposed to a link-level metric, and (2) we perform passive measurements directly from the broadcast control traffic injected by the protocol under test as it is running. Our approach is particularly valuable for unstable topologies that show different behaviors at different times. Initially, we also attempted to measure the network before and/or after running the protocol, and for unstable topologies the Expected Network Delivery often appeared to be uncorrelated from the various performance dimensions.

Similarly to the CTP work [7] and to the aforementioned studies, we capitalize on remote-access testbeds and their backchannels to gain a thorough understanding of the reasons for packet loss. Similarly to the Visibility framework [22], our approach makes it easier to diagnose the causes of failures. Differently from that framework, however, our approach is unobtrusive because it does not require any changes to the protocol under test. Similarly to [8], we note that testbed conditions vary so rapidly that even back-to-back experiments are not guaranteed to share the same conditions, which is why we measure the PRR from a protocol's control traffic while the protocol is running.

While in [22] visibility is pursued from within the protocol, the achievement of visibility through passive inspection by way of a sniffer network is the focus of [14], [16], and [15]. We believe that coupling our method with passive inspection techniques would greatly benefit the overall system visibility that passive inspection strives for.

This work is informed with a deep awareness of the vagaries of wireless propagation [17], in particular the existence of the transitional region of connectivity [24][25] and the temporal properties of wireless links [3].

5 Conclusion

The wide range of protocol performance levels across different topologies suggests that just looking at the performance results with no regard to the topology only gives an incomplete picture of the protocol performance. The END is a significant step towards a systematic methodology for the comparison of experimental results across protocols, time, and testbeds. We showed that the END exposes specific features of the network topology that can significantly affect the network performance.

The effectiveness of our approach in describing the state of the network during an experiment suggests that it is possible to succinctly represent the network topology in the context of the design goals of a protocol. We primarily focused on collection protocols, but also showed that our methodology applies to point-to-point routing. We underscored its added value in the context of testbed experiments, and we showed that our framework is also applicable to the characterization of simulation scenarios.

Acknowledgements. This work was partially supported by the European Commission under the SCAMPI Project (ICT grant agreement 258414), the Stanford Army High Performance Computing Research Center (grant W911NF-07-2-0027), and the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS) (Swiss National Science Foundation grant number 51NF40-130758/1).

References

1. Castalia - A Simulator for WSNs, `http://castalia.npc.nicta.com.au`
2. Bathula, M., Ramezanali, M., Pradhan, I., Patel, N., Gotschall, J.: A sensor network system for measuring traffic in short-term construction work zones. In: Krishnamachari, B., Suri, S., Heinzelman, W., Mitra, U. (eds.) DCOSS 2009. LNCS, vol. 5516, pp. 216–230. Springer, Heidelberg (2009)
3. Cerpa, A., Wong, J., Potkonjak, M., Estrin, D.: Temporal Properties of Low Power Wireless Links: Modeling and Implications on Multi-Hop Routing. In: ACM/IEEE Fourth International Symposium on Information Processing in Sensor Networks (IPSN 2005), Los Angeles, CA (April 2005)
4. Chakeres, I., Royer, E., Perkins, C.: Dynamic MANET On-demand Routing Protocol. IETF Internet Draft – work in progress draft-ietf-manet-dymo-00 (February 2005)
5. Dijkstra, E.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik 1, 269–271 (1959)

6. Faloutsos, M., Faloutsos, P., Faloutsos, C.: What does the Internet look like? Empirical Laws of the Internet Topology. In: SIGCOMM 1999, Cambridge, MA, USA (September 1999)
7. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection Tree Protocol. In: 7th ACM Conference on Embedded Networked Sensor Systems (SenSys 2009), Berkeley, CA (November 2009)
8. Gnawali, O., Guibas, L., Levis, P.: A Case for Evaluating Sensor Network Protocols Concurrently. In: The Fifth ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and Characterization (WINTECH 2010), Chicago, IL, USA (September 2010)
9. Ko, J., Gao, T., Terzis, A.: Empirical Study of a Medical Sensor Application in an Urban Emergency Department. In: 4th Intl Conference on Body Area Networks (BodyNets 2009), Los Angeles, CA (April 2009)
10. Levis, P., Lee, N., Welsh, M., Culler, D.: TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In: 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), Los Angeles, CA, USA (November 2003)
11. Puccinelli, D., Haenggi, M.: Spatial Diversity Benefits by Means of Induced Fading. In: Third IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON 2006), Reston, VA, USA (September 2006)
12. Puccinelli, D., Haenggi, M.: Reliable Data Delivery in Large-Scale Low-Power Sensor Networks. ACM Transactions on Sensor Networks (July 2010)
13. Radoslavov, P., Tangmunarunkit, H., Yu, H., Govindan, R., Shenker, S., Estrin, D.: On Characterizing Network Topologies and Analyzing Their Impact on Protocol Design. Tech. Rep. 00-731, University of Southern California (February 2000)
14. Ringwald, M., Römer, K., Vitaletti, A.: Passive Inspection of Sensor Networks. In: Aspnes, J., Scheideler, C., Arora, A., Madden, S. (eds.) DCOSS 2007. LNCS, vol. 4549, pp. 205–222. Springer, Heidelberg (2007)
15. Roemer, K., Ma, J.: Pda: Passive distributed assertions for sensor networks. In: 8th International Conference on Information Processing in Sensor Networks (IPSN 2009). IEEE Computer Society, San Francisco (2009)
16. Roemer, K., Ringwald, M.: Increasing the Visibility of Sensor Networks with Passive Distributed Assertions. In: Workshop on Real-World Wireless Sensor Networks (REALWSN 2008), Glasgow, UK (April 2008)
17. Srinivasan, K., Dutta, P., Tavakoli, A., Levis, P.: An Empirical Study of Low-Power Wireless. ACM Transactions on Sensor Networks (to appear, 2010)
18. Srinivasan, K., Jain, M., Choi, J., Azim, T., Kim, E.: The Kappa Factor: Inferring Protocol Performance Using Inter-link Reception Correlation. Tech. Rep. 09-02, Stanford University (2009)
19. Srinivasan, K., Kazandjieva, M., Agarwal, S., Levis, P.: The Beta-Factor: Improving Bimodal Wireless Networks. In: 6th ACM Conference on Embedded Networked Sensor Systems (SenSys 2007), Raleigh, NC (November 2008)
20. Srinivasan, K., Kazandjieva, M., Jain, M., Kim, E., Levis, P.: SWAT: Know Your Network. In: 8th International Conference on Information Processing in Sensor Networks (IPSN 2009), San Francisco, CA (April 2009)
21. Colesanti, U., Santini, S.: A Performance Evaluation Of The Collection Tree Protocol Based On Its Implementation For The Castalia Wireless Sensor Networks Simulator. Tech. Rep. 681, ETH Zurich (August 2010)
22. Wachs, M., Choi, J., Lee, J., Srinivasan, K., Chen, Z., Jain, M., Levis, P.: Visibility: A New Metric for Protocol Design. In: 5th ACM Conference on Embedded Networked Sensor Systems (SenSys 2007), Sydney, Australia (November 2007)

23. Werner-Allen, G., Swieskowski, P., Welsh, M.: MoteLab: a Wireless Sensor Network Testbed. In: 4th International Symposium on Information Processing in Sensor Networks (IPSN 2005), Los Angeles, CA (April 2005)
24. Woo, A., Tong, T., Culler, D.: Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In: 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), Los Angeles, CA (November 2003)
25. Zuniga, M., Krishnamachari, B.: An Analysis of Unreliability and Asymmetry in Low-Power Wireless Links. ACM Transactions on Sensor Networks 3(2), 1–30 (2007)

An Adaptive Algorithm for Compressive Approximation of Trajectory (AACAT) for Delay Tolerant Networks

Rajib Rana[1,2], Wen Hu[2], Tim Wark[2], and Chun Tung Chou[1]

[1] School of Comp. Sci. and Engineering, University of New South Wales, Australia
{rajibr,ctchou}@cse.unsw.edu.au
[2] CSIRO ICT Center, Australia
{lastname.firstname}@csiro.au

Abstract. Highly efficient compression provides a promising approach to address the transmission and computation challenges imposed by moving object tracking applications on resource constrained Wireless Sensor Networks (WSNs). In this paper, we propose and design a Compressive Sensing (CS) based trajectory approximation algorithm, *Adaptive Algorithm for Compressive Approximation of Trajectory (AACAT)*, which performs trajectory compression, so as to maximize the information about the trajectory subject to limited bandwidth. Our extensive evaluation using "real" trajectories of three different object groups (animals, pedestrians and vehicles) shows that CS-based trajectory compression reduces up to *30% transmission overheads*, for given information loss bounds, compared to the state-of-the-art trajectory compression algorithms. We implement AACAT on the resource-impoverished sensor nodes, which shows that AACAT achieves high compression performance with very limited resource (computation power and energy) overheads.

1 Introduction

Object tracking is on horizon due to multitude of application scenarios in the present time. The Virtual Fencing (VF) application devised by the CSIRO ICT Center, Australia is one such example, where the locations of animals are controlled, not by a physical fence, but with stimuli (e.g. auditory and mild electric shocks) applied by special devices worn by the animals. Ethical considerations are critical in the VF application and observations regarding the states of each animal must be maintained. Since continuous human observation is infeasible, it is important to return data about the states of the animals and the stimuli applied. The VF application operates in a delay tolerant fashion. The position data of an animal is recorded and stored in a sensor node attached to the animal. There is also a small number of fixed nodes which are connected to the basestations. When animals enter the transmission range of a fixed node, position data from their sensor nodes are uploaded to the basestations. Due to uncoordinated movement of the animal connection time (amount of time a fixed node is connected with the mobile node) is typically unpredictable. Therefore, the amount of data that can be uploaded during a connection is also unpredictable in the VF application.

The VF application poses two fundamental challenges. First, monitoring a large number of animals requires the availability of their complete geographical traces (trajectories), which leads to *transmission challenges* due to limited bandwidth of wireless nodes. Second, limited transmission opportunities and connection time often cause the memory buffer to become full, therefore it becomes necessary to discard data, which leads to data loss.

Besides the VF application, trajectory compression is becoming essential in a number of participatory sensing [2] applications, such as, social networking (user locations need to be continuously uploaded) and traffic conditions monitoring (traffic condition is inferred by analyzing position data uploaded from vehicles) using mobile phones. Although, mobile phones have access to higher bandwidth, expensive cellular communications often restricts the amount of data that can be transmitted. GPRS or cellular bandwidth imposes strong restriction on the amount of transmission.

An efficient trajectory compression algorithm helps to cope with limited bandwidth; therefore, trajectory compression has been extensively studied in the last decades. One class of these compression algorithms [6,12] is mainly guided by the advances in the field of line simplification and cartographic generalization. The primary disadvantage of these algorithms is the frequent elimination or misrepresentation of important points, such as sharp angles. Another class [1,10] of algorithms achieves compression via predictions; however, these algorithms generally assume that the mobile objects have a limited number of movement states (e.g moving speed and direction), which may be impractical in the reality.

Recent developments in Compressive Sensing (CS) theory [3] provide an attractive alternative for trajectory compression in WSNs. Given that a trajectory segment $f \in \mathbb{R}^n$ is compressible in a sparsifying domain (e.g. Fourier, Discrete Cosine Transform (DCT) etc.), then a small number of coefficients (which will be denoted by k is this paper), is sufficient to *accurately* represent the trajectory segment. The key idea behind CS is that it takes a small number of projections $m(<< n)$ to *accurately* recover f with high probabilities (projections are typically aggregations of data points of a trajectory segment, we define it in detail in Section 3). Since, only $m(<< n)$ projections need to be transmitted to the basestations, CS offers efficient use of bandwidth.

The key challenge of applying CS in trajectory compression is that we need to estimate the compressibility, namely the parameter k, of the trajectory in-situ to obtain the most benefits out of CS in practice, because the number of compressive projections (m) is a function of compressibility (k) that changes dynamically with the speed of the mobile objects. Intuitively, when an object is stationary or moves with a constant speed, the compressibility of its trajectory is high and thus it requires a smaller number of projections to recover the trajectory accurately. On the other hand, when the speed of the object changes frequently, the compressibility of the trajectory diminishes and it requires a larger number of projections to recover the trajectory accurately. However, the conventional method of computing compressibility is computationally expensive, and is infeasible for resource constrained WSN nodes. We propose and design a Support Vector Regression (SVR), namely *ϵ-SV regression* [18], based in-situ compressibility estimation technique, to provide an accurate estimation of k based on the speed information of the nodes, and has small computational overheads.

Our key contributions can be summarized as follows:

1. We present an *Adaptive Algorithm for Compressive Approximation of Trajectory (AACAT)*, which adapts the number of CS projections while accurately approximating the trajectory. An ϵ-SV regression based estimation is proposed to adapt the number of CS projections to the object's speed in-situ, which improves the compression performance of CS based on the local speed observations.
2. Our evaluations, based on the trajectories of three types of moving objects, exhibiting large range of speed variations, show that CS-based trajectory compression achieves *30%* better "transmission versus accuracy trade-off", compared to two other state-of-the-art trajectory compression algorithms based on *Kalman filter* and *Spatiotemporal* compression techniques. Our evaluations also show that CS-based trajectory compression is resilient to information loss as it offers promising "loss-distortion trade-off".
3. Our *implementation* of AACAT on Fleck 3b platform consisting of an 8-bit microcontroller and 8 kB RAM, demonstrates the resource efficiency of AACAT that makes it a viable algorithm for resource impoverished sensor nodes. Furthermore, we show that AACAT approximates trajectories with high accuracy using empirical study.

2 Datasets

We evaluate the performance of AACAT using three datasets from WSN deployments and field experiments. These three datasets are made up of animal (cow), pedestrian and vehicle trajectories respectively, which will demonstrate the performance of AACAT with a large range of speed variation of mobile nodes. The animal trajectories were collected from a real WSN deployment at the Wivenhoe Dam, where sensor nodes are mounted on the cow collars and locations are sampled by the GPS receivers at 2 Hz. We collected trajectories of *36 cows* for one week. The pedestrian and vehicle datasets were created with the help of *20 staff* and *student members* of the CSIRO ICT center. They were given Nokia N97 mobile phones for a two-week period to use during their daily commute. Annotation was used to differentiate vehicle and pedestrian data. A python script was installed on the phone that recorded the GPS coordinates at 2 Hz. In summary, we collected 75,000 segments of cow trajectories, 1,000 segments of pedestrian trajectories and 5,000 segments of vehicle trajectories, where each segment has $n = 512$ data points[1].

[1] We also conducted experiments with smaller segment lengths e.g. 128, 256 and observed similar results.

The ϵ-SV regression method used for estimating k involves a training and a test phase. We divided our datasets into training and testing datasets to use during the training and validation phases, respectively, and ensured that they are independent. For example, the training set for pedestrian was formed using the trajectories within the campus of the CSIRO ICT center and the test set was formed using trajectories outside of the CSIRO ICT center campus. For animals, we used the trajectories of half of the cows as the training set and trajectories from the rest of the cows as the test dataset. Finally, training and test datasets for vehicle were formed using trajectories from separate road segments.

3 Compressive Sensing for Trajectory Compression

We first present the problem that we solve in this paper followed by the basic idea of CS and how to apply CS theory to trajectory compression.

3.1 Problem Formulation

Consider $f \in R^n$ is a trajectory segment containing n consecutive position data points of a moving object. In order to conserve bandwidth, we want to acquire only $m << n$ projections of f, which can be used to accurately reconstruct f at the basestations. Due to limited transmission opportunities and connection time, some of these m projections may further be discarded. Therefore, it is also required that reconstruction performance degrades gracefully with the loss of projections.

3.2 Compressive Sensing

The theory of CS provides an attractive solution to the problem of recovering a compressible signal from a few projections. Given the trajectory segment f is an n data point discrete time signal, using an $n \times n$ orthonormal basis matrix $\Psi = [\psi_1|\psi_2|...|\psi_n]$ with the vectors ψ_i as basis vectors, f can be represented as,

$$f = \Sigma_{i=1}^{n} \psi_i \alpha_i \text{ or } f = \Psi \alpha, \tag{1}$$

where, α is an $n \times 1$ column vector of weighting coefficients $\alpha_i = \langle f, \psi_i \rangle = \psi_i^T f$ and $.^T$ denotes the transposition. Note that a position data point is typically represented using two geographic Cartesian coordinates, Northing and Easting, where Northing refers to the northward-measured distance (or the y-coordinates) and Easting refers to eastward-measured distance (or the x-coordinates). For simplicity, we consider Northing and Easting separately i.e. f contains either n consecutive Northing or Easting data points. Our approach is to separately recover the Northing and Easting data points of a given trajectory segment to recover the corresponding segment. Note that f and α are equivalent representation of the trajectory segment, with f in the time domain and α in the Ψ domain. Generally, f is compressible when it has a few large and many small coefficients in the Ψ domain. Formally, f is compressible when the reordered entries of its Ψ-coefficients decay like power law; i.e. when we rearrange the sequence of α in decreasing order of magnitude $|\alpha|_{(1)} \geq |\alpha|_{(2)} \geq ... \geq |\alpha|_{(n)}$, for some the $r \geq 1$, the zth largest entry obeys,

$$|\alpha|_{(z)} \leq \text{Const.} z^{-r}. \tag{2}$$

Instead of conducting *point-wise* measurements, CS takes $m << n$ *projections* of f, where each projection is an inner product between f and a projection vector ϕ_j as in $y_j = \langle f, \phi_j \rangle$. Forming a $m \times n$ projection matrix Φ using m vectors ϕ_j^T, we can write the projection operation in matrix form, as follows:

$$y = \Phi f = \Phi\Psi\alpha = \Theta\alpha \tag{3}$$

Eq. (3) represents the typical CS *encoding process*, where $\Theta = \Phi\Psi$.

Since $m << n$, the problem of recovering f from y and Θ is ill-conditioned. CS shows that a sufficient condition for a stable solution is that for an arbitrary $3k$ sparse vector v and for some $\delta > 0$, Θ satisfies

$$1 - \delta \leq \frac{||\Theta v||}{||v||} \leq 1 + \delta. \tag{4}$$

Condition (4) is called Uniform Uncertainty Principle (UUP) [4] or Restricted Isometry Property (RIP). CS theory also suggests mechanisms to generate Θ matrices that satisfy the RIP with high probabilities. For example, if Φ is formed by sampling iid entries from the normal distribution with mean 0 and variance $1/m$, then for an arbitrary orthonormal basis Ψ, $\Theta = \Phi\Psi$ obeys RIP with overwhelming probability, when $m \geq ck \log\left(\frac{n}{k}\right)$ with c as a small constant [5].

Condition (4) ensures that f can be recovered from (3); however, since $m << n$, there are many α' that satisfies $y = \Theta\alpha'$. Encouragingly, CS optimization based on ℓ_1 norm

$$\hat{\alpha} = \arg\min_{\alpha'} ||\alpha'||, \text{ s.t. } y = \Theta\alpha' \tag{5}$$

can accurately estimate f (namely producing the k-term approximation by returning its largest k coefficients) with high probabilities, using only $m \geq ck \log\left(\frac{n}{k}\right)$ projections [4]. Eq. (5) is the CS *decoding (reconstruction)* process. Since (5) is a convex optimization problem, it can be solved in polynomial time and k-approximation of f (which will be denoted by $\hat{f}$) can be recovered using $\hat{f} = \Psi\hat{\alpha}$.

Given that f is compressible, i.e. k is very small, we have $ck \log\left(\frac{n}{k}\right) << n$. Therefore, using the CS theory, we can accurately recover f using only a small number of projections m and achieve high compression ratio $\frac{m}{n}$, since $m << n$.

Consider, we compute $m (= ck \log n)$ projections of f. Due to limited transmission opportunities and connection time, some of these projections (e.g. μ) are discarded and only $\hat{m} = m - \mu$ of them are transmitted to the basestations. Using $\hat{m}$ projections we cannot recover f accurately, rather we can produce $\hat{k} \leq k$ approximation of f with an increase of the reconstruction error. However, in Section 6.1 we will empirically show that the reconstruction error increases gracefully with the increase of μ.

3.3 Adaptive Compressive Sensing

In order to get the most benefit out of CS in practice, we need to find a basis where the trajectory data points are most compressible. In [15] we report that trajectories of all three object groups have most compressible representation in DCT. Now consider that a trajectory segment has k significant coefficients while represented in DCT. In CS, the number of projections, m, to approximate the trajectory segment accurately is determined by this number of significant coefficients, k. Typically, k changes dynamically with the speed of the object. Therefore, computing k for every trajectory segment would further improve the compression performance. However, computing k for a trajectory segment involves a transpose of an $n \times n$ matrix ($O(\frac{n^2}{2})$ operations) and then multiplication of the $n \times n$ matrix with an $n \times 1$ vector ($O(n^2)$ operations), which are computationally expensive for resource impoverished wireless sensor nodes. Note that the transpose operation can be saved by storing Ψ^T in the memory, however the multiplication ($\Psi^T f$) will still incur high computational expenses. An efficient estimation of k is therefore required, which accurately estimates k incurring affordable computation cost. In the next section we will propose an ϵ-SV regression based technique to estimate k, which accurately estimates k incurring reasonable computational expenses.

4 In-situ Estimation of *k*

In this section we first show the correlation of the speed of the moving objects with k, then we use ϵ-SV regression to model this correlation.

4.1 Correlation between Speed and *k*

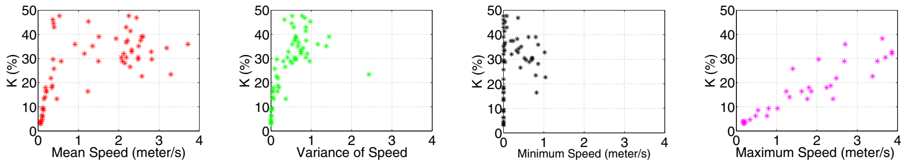
Given $f \in \mathbb{R}^n$ is an n data point Northing or Easting segment of a moving object recorded over time $t_i, (i = 1, ..., n)$, the quantity $\frac{|f_{i+1}-f_i|}{t_{i+1}-t_i}, (i = 1, ..., n-1)$ produces the speed along the corresponding Northing or Easting axis. We aim to estimate k based on the speed of the moving object.

We process the trajectory in the test data set, segment by segment where each segment is of length n. The n data points of a trajectory segment gives $n-1$ speed readings along both Northing and Easting axes. In order to investigate the correlation of speed with k, for each segment we compute mean, variance, minimum and maximum (we call them *speed candidates* hereafter) of the corresponding $n-1$ speed readings and plot them against the corresponding k values, where we use a value of k that approximates the corresponding trajectory segment (Northing and Easting separately) within a small (one meter) error.

Figure 1 summarizes the correlations of different speed candidates with k over all the segments of the training set from animals[2]. For a given speed candidate

[2] Due to space constraints, in Fig. 1 we only depict the correlation for Northing. Similar results are observed for Easting. Furthermore, we only use the animal dataset for this illustration since similar to animal, for the rest two datasets, mean, variance and maximum speed show the best correlation with k.

e.g mean speed, we compute the mean speed of each of the trajectory segments and compute the corresponding k value to approximate the trajectory segment within one meter error. We then plot these mean speeds along x-axis and corresponding k values in y axis in Fig. 1(a). In particular, instead of plotting the k values, we plot the ratio between k and the total number of coefficients within a trajectory segment along the y-axis. We observe that *mean*, *variance* and *maximum* speed are well correlated with k, because with the increase of these speed candidates, the corresponding values of k increase proportionally. For clarity we also show the value of correlation coefficients (CC) in the caption of the corresponding figure. In the next section we will model the correlations between the three chosen speed candidates (mean, variance and maximum) and k using ϵ-SV regression.



(a) CC=0.355 (b) CC=0.322 (c) CC=0.008 (d) CC=0.459

Fig. 1. The correlations of various speed candidates with k. The values of the correlation coefficients (CC) are shown in the caption of corresponding figure.

4.2 Modeling Correlation by ϵ-SV Regression

Consider for a given speed candidate s, the training set,$\{(s_1, k_1), ..., (s_L, k_L)\}$, $s \in \mathbb{R}, k \in \mathbb{R}$, has L elements, where s_i is ith value of s and k_i is the corresponding value of k. The ϵ-SV regression determines a function $g(s)$ that has at most ϵ deviation from the actual k_i for all the training data, and at the same time is as flat as possible. The function g is typically computed from

$$g(s) = \langle \omega, s \rangle + b \text{ with } \omega \in \mathbb{R}^n, b \in \mathbb{R}, \tag{6}$$

where $\langle ., . \rangle$ denotes the dot product within $\mathbb{R}^n$. In (6) *flatness* is achieved by *minimizing* ω. Because of space limit, we only show the case for linear function in (6), readers are encouraged to read [18] for the complete description of ϵ-SV regression. One of the main characteristics of ϵ-SV regression is that it is a Quadratic Problem (QP) which has a unique global solution in general.

In Section 4.1, we have shown that three candidates namely mean, variance and maximum values of speed have close correlation with k. Using ϵ-SV regression we further attempt to find the best candidate among them. We use the combinations of these candidates to increase the scope of our search. We first train the ϵ-SV regression process using the mapping of k with different speed candidates of the *training dataset*. In these mappings the values of k were chosen to approximate the corresponding trajectory segment within one meter error[3].

[3] We chose the values of k to approximate Northing and Easting separately within 0.5 meter, therefore the resultant distance error is less than or equals to one meter.

Table 1. MEE of k estimation for different speed candidates. The smallest MEE for different forms of the datasets are highlighted in the corresponding column.

Speed candidates	MEE of k estimation			
	animal	vehicle	pedestrian	universal
mean	2.97	1.28	0.19	3.82
variance	**2.74**	2.06	**0.19**	**3.29**
max	3.67	1.05	0.20	4.41
mean, variance	2.78	1.29	0.19	3.47
mean, max	3.44	0.93	0.19	4.35
variance, max	3.10	1.05	0.19	3.95
variance, mean, max	3.25	**0.92**	0.19	4.16

For each speed candidate, the training process computes the correlation function $g(s)$ as in (6).

We then use the *test datasets* to estimate k for different candidates by passing the values of the speed candidate (s) to the function $g(s)$ computed during the training process. We also use two different forms of datasets within the training and validation process. In the first form, we use the data from individual object group (i.e. animal, pedestrian separately), whilst in the second form we combine data from all three object groups to form a "universal" dataset. In particular, we combine the training and test datasets of individual object group to form, respectively, the universal training and test datasets. The underlying idea of using a universal dataset is to investigate the performance of a universal training (training SVR using the combined datasets of different object groups) over individual training (training SVR using individual object group dataset). Note that a universal training can lessen the requirement of using individual training results for estimating k of corresponding object trajectories.

We compute the Mean Estimation Error (MEE) to compare the the performance of estimating k using different speed candidates. If there are total J segments in a trajectory dataset, MEE is computed by $(\frac{1}{J}\sum_{i=1}^{J}(\delta_E^i+\delta_N^i))$, where δ_E^i is the absolute difference between the test and the estimated k for ith trajectory segment (Easting) and δ_N^i is the absolute difference between the test and the estimated k for ith trajectory segment (Northing). MEE for various candidates are summarized in Table 1. We observe that for the animal, pedestrian and universal datasets, the variance of speed produces the smallest MEE, but the combination of mean, variance and maximum speed produces the smallest MEE for the vehicle dataset.

In order to use ϵ-SV regression for estimating k in-situ, we create a lookup table (ϵ-SV lookup) based on the ϵ-SV regression training results. The advantage of using a lookup table over running resource-intensive ϵ-SV regression on the nodes is the higher computation efficiency, which is crucial for resource impoverished sensor nodes. The disadvantage of this strategy is relatively lower quality estimation of k, because of the limitation of search spaces of a pre-calculated lookup table can have. We will demonstrate this design trade-off in details in Section 6.

5 AACAT Algorithm

In our current implementation, AACAT has two main threads where Thread 1 stores and compresses the data (generates projections) and Thread 2 opportunistically transmits the projections to the basestations. Memory management is handled via fixed-length linked lists, where we store the position data and the projections in two separate lists called `receiveQUEUE` and `sendQUEUE`, respectively (for Northing and Easting we use separate `sendQUEUE` and `receiveQUEUE`). The size of the `receiveQUEUE` is chosen so that it can fit data of one trajectory segment. Each segment is identified using the timestamp of the first sample in the segment. Projections attached with the same timestamp are used to reconstruct the corresponding segment in the basestations. Northing and Easting segments are reconstructed separately and then combined to recover the corresponding trajectory segment. We also use separate lookup tables for Northing and Easting, where the values of the speed candidate(s) are rounded to one decimal place.

Thread 1 constantly takes samples from the GPS module and adds the samples to the `receiveQUEUE` when the `receiveQUEUE` is not full. Once the `receiveQUEUE` is full, Thread 1 computes the projection(s) of the samples in the queue, by first determining k using the ϵ-SV lookup table, followed by computing $m = \lceil ck \log\left(\frac{n}{k}\right)\rceil$ projections of the samples. We observed $c = 1$ is sufficient for pedestrian and vehicle datasets, but a higher value ($c = 2$) is required for animal dataset. We used iid Gaussian $\mathcal{N}(0, \frac{1}{m})$ numbers as the elements of the projection vector which can be constructed distributedly at the node and the basestations by using the same seed of a pseudo-random generator. The lookup operation selects the value of the speed candidate in the table that has the smallest difference with the speed candidate of the current segment and retrieves the corresponding k value.

If the `sendQUEUE` is not full, projections are appended in the `sendQUEUE`. Otherwise, the projections can be discarded from the `sendQUEUE` randomly because each projection is a linear combination of the samples in a segment. This property offers promising loss-distortion trade-off as in Section 6 we demonstrate that the reconstruction error increases gracefully as we reduce the number of projections. Algorithm 1 shows the pseudocode of Thread 1.

The role of Thread 2 is to detect the network connectivity. Once the network is connected, it attempts to transmit a copy of the element at the head of `sendQUEUE`. If the element is successfully received, it removes the sample at the head of the list that in turns frees up one space in the `sendQUEUE`.

Algorithm 1. Thread 1

```
Ensure: Memory for linked list is allocated
 1: receiveQUEUE ← list to store incoming position data (GPS samples)
 2: sendQUEUE ← list to store outgoing AACAT projections
 3: while GPS has lock do
 4:    p ← New GPS position
 5:    if receiveQUEUE NOT full then
 6:       push p to receiveQUEUE
 7:    else
 8:       look up $k$ of the segment at receiveQUEUE and compute $m = \lceil ck \log(\frac{n}{k}) \rceil$
          projections.
 9:       for all projection do
10:          if sendQUEUE NOT full then
11:             push one projection in sendQUEUE.
12:          else
13:             if Each segment has one projection then
14:                Randomly delete one segment.
15:             else
16:                Randomly delete one projection of a segment having more than one
                   projection.
17:             end if
18:          end if
19:       end for
20:    end if
21: end while
```

6 Evaluation

We use Average Distance Error (ADE), which gives the distance between the real and reconstructed trajectory.

Consider that $N_\omega^i, 1 \leq i \leq n$ and $E_\omega^i, 1 \leq i \leq n$ are respectively the consecutive Northing and Easting data points of a trajectory segment ω, and $\hat{N}_\omega^i, 1 \leq i \leq n$ and $\hat{E}_\omega^i, 1 \leq i \leq n$ are the reconstruction of $N_\omega^i, 1 \leq i \leq n$ and $E_\omega^i, 1 \leq i \leq n$, respectively. If there are J segments in a trajectory dataset, we compute ADE by,

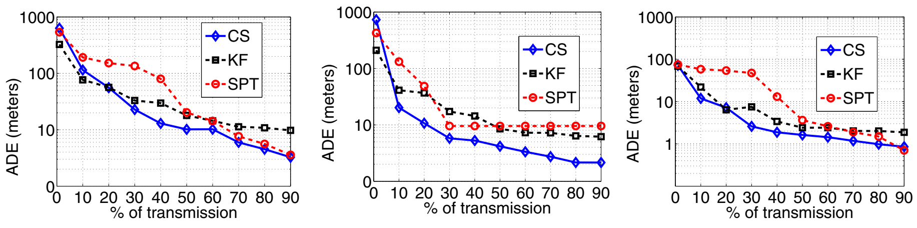
$$\text{ADE}=\frac{1}{nJ}\sum_{w=1}^{J}\sum_{i=1}^{n}\sqrt{(N_\omega^i-\hat{N}_\omega^i)^2+(E_\omega^i-\hat{E}_\omega^i)^2}.$$

6.1 Transmission–Accuracy Trade-off of CS

CS achieves promising transmission versus accuracy trade-off, since it requires only a small number of projections to accurately recover a trajectory segment. We compare this trade-off of CS with that of two widely used trajectory compression algorithms: *Kalman Filter (KF)* and *Spatiotemproal* compression algorithm (SPT).

KF is a stochastic, recursive data filtering algorithm widely used for system state estimation, where state estimation process operates using recursive steps of prediction and correction based on observations. We used a centralized implementation of KF where Northing and Easting were separately modeled as state variables and the predictions were made remotely at the basestations using the data acquired from mobile nodes. In order to make a fair comparison the number of bytes transmitted by CS is the same as the number of bytes transmitted by KF (Note that the number of bytes (4kB) for a projection value is the same as the number of bytes for a Northing or Easting point).

SPT is the improvement proposed by Meratina et al. [12] to the prominent Douglas-Peucker (DP) [7] method. It leverages the Synchronous Euclidean Distance (SED) [13] that is the measurement of the distance between an observed position and its estimated position based on a constant velocity model. For a triplet of points with low SED, the middle point can be removed from the trajectory with small loss of information. Similar to KF, the number of bytes transmitted by SPT is the same as the number of bytes transmitted by CS. Note that in both SPT and KF, when there is a missing data point, we use the last observed data point as the current data point.



(a) Animal trajectory (b) Vehicle trajectory (c) Pedestrian trajectory

Fig. 2. Benchmarking CS using KF and SPT

In Fig. 2 we compare the transmission-accuracy trade-off of CS, KF and SPT. For CS, along x-axis is the percentage of the number of projections to the total number of data points, but for KF and SPT, along x-axis is the percentage ratio of the number of transmitted data points to the total number of data points. For all CS, KF and SPT along y-axis is the ADE. We consider an n data point trajectory segment which is comprised of n Northing and n Easting data points. Therefore, the total number of data points within a trajectory segment is $2n$, and the number of transmitted projections or data points is the summation of the number of transmitted Northing and Easting projections or data points, respectively. We observe that, to approximate an animal trajectory segment within 10 m error, both SPT and KF requires approximately 30% more transmissions compared to CS. However, due to higher compressibility, the difference between CS and KF/SPT is smaller for the vehicle and pedestrian trajectories.

Due to delay tolerant transmission and fixed size memory buffer, data loss is very likely to happen in the delay tolerant networks. We illustrate some examples of possible data loss in Fig. 3, where for different latencies along the x-axis we compute the percentage of possible transmissions along y-axis. Consider a

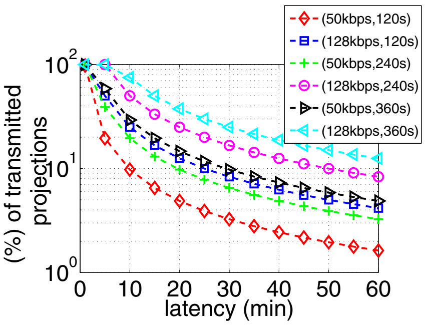


Fig. 3. Latency versus transmission

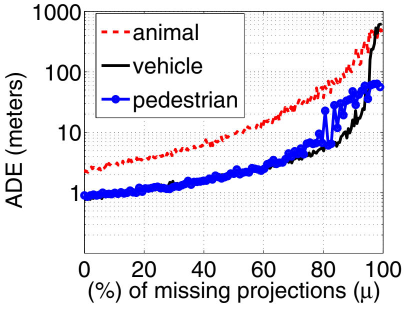


Fig. 4. Loss distortion trade-offs

sampling frequency h Hz and inter-connection interval (since a mobile node has only intermittent connection with the fixed node, the inter-connection interval is the time between 2 connections with the fixed node by a mobile node) ℓ seconds. The total number of projections need to be transmitted during ℓ seconds is: $T = 0.4\ell h$ (we arbitrarily choose 0.4, since from Fig. 2, 40% transmission approximates the animal trajectory within 10 m error). Then, for a bandwidth B kbps and connection time τ seconds (the amount of time a mobile node associates with a fixed node), the allowed transmission is $\Delta = \tau B$ kb. The percentage of CS transmission is therefore $\frac{\Delta}{TQ}$, where each projection is Q kb. In Fig 2, we use three different connection times, 120, 240 and 360 s, respectively and two different bandwidths, 50 and 128 kbps, respectively. We observe that unless the inter-connection interval is very small (1 or 2 min), even for high bandwidth (e.g. 128 kbps) and connection time (e.g. 360 s), it is impossible to transfer 100% projections.

One key advantage of the CS theory is that it offers promising loss-distortion trade-offs. In Fig. 4 we illustrate the loss distortion trade-off of CS based reconstruction. Along the x-axis is the percentage of missing projections and along the y-axis is the corresponding distance error. We first compute the average number of projections (η) required to approximate a trajectory segment within one meter error. Starting from η, we gradually reduce the number of projections, compute the corresponding error, and plot the results in Fig. 4. We observe that reconstruction performance degrades gracefully with the loss of projections. Such as, for all three object groups, for up to 60% loss of projections, the ADEs are within 10 meters. However, the reconstruction error increases rapidly (especially for animals) beyond that point.

We have made a number of comparisons to evaluate the performance of AACAT. Due to space limit we are unable to report the results here. Reader are encouraged to read [15] to find the detailed about these comparison results.

6.2 System Performance

We implemented AACAT on the Fleck 3b nodes, which are in-house sensor platform based on an 8-bit Atmel Amega 1281 microcontroller with 8 kB RAM

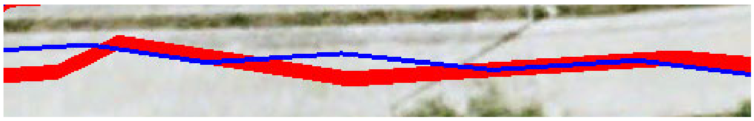


Fig. 5. The reconstruction performance of AACAT on Fleck 3b platform. The thick line is the ground truth and the thin line is the trajectory approximation produced by AACAT. ADE is 3.67 meters.

Table 2. Memory and energy consumption of AACAT

Process	current (mA)	time (ms)	energy (mJ)	memory (kB)
projection	8	16.4 ± 0.8955	0.47	1.18
lookup	8	$7.68 \pm .01$	0.18	1.67

(similar to Mica family motes) and a 50 kbps Nordic NRF905 transceiver working on ISM 900 band, and evaluated the system performance with field experiments. During the experiments, each subject (human) was given a Fleck to carry around and was asked to walk along a road segment outside our lab. A fixed (connected) node was placed in the middle of the road segment which was one hop away from a basestation placed in the lab.

Two receiveQUEUEs of capacity of 512 data points were declared, one for Northing and the other for Easting. Projected values of Northing and Easting were stored separately in two sendQUEUEs with capacity of 128 projections each. Size of the queues was chosen based on available memory within the 8 kB RAM total on a Fleck 3b. Beacons were broadcast from the fixed node every second. After hearing a beacon from the fixed node, mobile nodes transmitted packets which would be forwarded to the basestations. The reconstruction result of a trajectory segment is shown in Fig. 5. The thick line is the ground truth which was recorded by carrying an additional Fleck 3b, recording GPS samples at 2 Hz. It is evident that the reconstruction is very close (ADE is 3.67 meter) to the real trajectory.

The projection and lookup operations are two key operations in AACAT. In Table 2, we summarize the memory and energy usage of one ϵ-SV lookup and one projection operations of 512 data points. The mean computation time for both operations is small. For example, it takes only approximately 16.4 ms for a Fleck3b to compute a projection, which is $16.4 \div (256 * 1000) \approx 0.006\%$ of the total time to collect one segment of data. Furthermore, the memory usage (maximum 14% of available RAM) and the energy consumption (maximum 0.47 mJ) of the projection and lookup operations are quite low and are affordable on the resource constrained WSN nodes.

7 Related Work

Approximation of the mobile object trajectory using only partial information collected from sensor nodes has been widely studied in the past. The central themes of the previously proposed algorithms can be classified into two broad categories: prediction and compression.

Prediction can be made in both centralized and distributed fashion. In a centralized prediction technique (e.g. [8]) a basestation based on the information extracted from the movement history of an object, predicts its future movement states, which are then sent to the corresponding sensor nodes. If the prediction do not match with the sensor readings, the sensor nodes correct the basestation by sending their own readings. To save node to base communications of the centralized techniques, dual prediction techniques are proposed in [11,21] where the predictions take place distributedly at both sensor nodes and basestation, and updates are sent to the basestation only when the prediction error exceeds some given threshold.

Both the centralized and distributed prediction techniques use either individual or group movement history for prediction. The prediction with individual history (e.g. [19,21]) predicts the movement of an object from its own history. Considering in practice an arbitrary movement trajectory that an object may follow, the simple prediction models in the existing work results in poor prediction performances. The prediction with group history (e.g. [1,10]) categorizes the objects into groups, and the prediction of a moving object is made based on the history of all objects from the same group. Group history provides richer information about the object movements than the individual history, but, these techniques assume a limited number of movement states (e.g. moving speed and direction) that a moving object can have. However, in practice even within the same group, different objects may demonstrate different movement patterns at different times (e.g., morning, noon and night) and/or with different tasks (e.g., surveillance and disaster response).

Compression algorithm proposed in [20] performs recursive segmentation of the trajectory, until a trajectory segment can be modeled with an interpolation function with a small error. Compression is achieved by only transmitting the relevant parameters of the interpolation function. However, compression performance of [20] has so far been evaluated using simulated trajectories without considering the resource (computation power, energy and bandwidth) usage efficiency, which is crucial in tiny embedded sensor nodes.

In [17] authors propose a low-energy adaptation of the lossless compression algorithm (LZW) for WSN, however, a likely scenario in the VF application is that in order to cope with limited bandwidth a large amount of data may need to be discarded. Therefore, a lossless compression algorithm is not a good fit. Furthermore, the authors do not exploit temporal correlations of the data to achieve compression, which can be explored to achieve better compression ratio.

Theoretical results provided in [9] show that CS is not an efficient compression technique while applied with ordinary quantization, however our empirical results show that CS provides reasonably good compression. In particular, we show that CS provides better compression compared to two other state-of-the-art trajectory compression techniques. Furthermore, in number of other papers [14,16] it is shown that CS provides promising compression in WSNs.

8 Conclusion

In this paper we present a trajectory approximation technique, called AACAT, which utilizes the embedded redundancy of trajectory data using the emerging theory of Compressive Sensing (CS). Furthermore, AACAT introduces a ϵ-SV regression-based in-situ compressibility estimation technique to adapt the number of required projections to trajectory data compressibility dynamically that improves the performance of CS trajectory compression. Our evaluation by three different trajectory datasets, collected by sensor nodes carried by animals, pedestrians and vehicles, shows that for a reasonable approximation accuracy, CS-based compression reduces 30% transmission overhead compared to the stat-of-the-art trajectory compression techniques driven by the classical Kalman Filter and Douglus-Peckur algorithms. Our evaluation also shows that CS-based trajectory compression is loss-resilient since the reconstruction error increases gracefully with the loss of projections. Finally, an end-to-end system, which is implemented on resource-impoverished sensor nodes with 8-bit microcontrollers and 8 kB RAM, demonstrates that AACAT achieves high compression performance with very little resource (computation power and energy) overhead.

References

1. Aslam, J., Butler, Z., Constantin, F., Crespi, V., Cybenko, G., Rus, D.: Tracking a moving object with a binary sensor network. In: SenSys 2003: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, pp. 150–161. ACM, New York (2003)
2. Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., Srivastava, M.B.: Participatory sensing. In: Workshop on World-Sensor-Web (WSW 2006): Mobile Device Centric Sensor Networks and Applications, pp. 117–134 (2006)
3. Candés, E.: Compressive sensing. In: Proc. of the Int. Congress of Mathematics (2006)
4. Candès, E.J., Tao, T.: Near-optimal signal recovery from random projections: Universal encoding strategies? IEEE Transactions on Information Theory 52(12), 5406–5425 (2006)
5. Candes, E.J., Wakin, M.B.: An introduction to compressive sampling [a sensing/sampling paradigm that goes against the common knowledge in data acquisition]. IEEE Signal Processing Magazine 25(2), 21–30 (2008), `http://dx.doi.org/10.1109/MSP.2007.914731`
6. Cao, H., Wolfson, O., Trajcevski, G.: Spatio-temporal data reduction with deterministic error bounds. In: DIALM-POMC 2003: Proceedings of the 2003 Joint Workshop on Foundations of Mobile Computing, pp. 33–42. ACM, New York (2003)
7. Douglas, D., Peucker, T.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Canadian Cartographer 10, 112–122 (1973)
8. Goel, S., Imielinski, T.: Prediction-based monitoring in sensor networks: taking lessons from mpeg. SIGCOMM Comput. Commun. Rev. 31(5), 82–98 (2001)

9. Goyal, V.K., Fletcher, A.K., Rangan, S.: Compressive sampling and lossy compression. IEEE Signal Processing Magazine 25(2), 48–56 (2008), http://dx.doi.org/10.1109/MSP.2007.915001
10. Ing, G., Coates, M.J.: Parallel particle filters for tracking in wireless sensor networks. In: Proceedings of the Signal Processing Advances in Wireless Communications, NY, USA (2005)
11. Jain, A., Chang, E.Y., Wang, Y.F.: Adaptive stream resource management using kalman filters. In: SIGMOD 2004: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 11–22. ACM, New York (2004)
12. Meratnia, N., de By, R.A.: Spatiotemporal compression techniques for moving point objects. In: Hwang, J., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 765–782. Springer, Heidelberg (2004)
13. Potamias, M., Patroumpas, K., Sellis, T.: Sampling trajectory streams with spatiotemporal criteria. In: SSDBM 2006: Proceedings of the 18th International Conference on Scientific and Statistical Database Management, pp. 275–284. IEEE Computer Society, Washington (2006)
14. Quer, G., et al.: On the interplay between routing and signal representation for compressive sensing in wireless sensor networks. In: Information Theory and Applications Workshop (ITA), San Diego, USA (January 2007), http://icapeople.epfl.ch/widmer/files/Tschopp2007Routing.pdf
15. Rana, R., Hu, W., Wark, T., Chou, C.T.: An adaptive algorithm for compressive approximation of trajectory (aacat) for delay tolerant networks. Tech. Rep. UNSW-CSE-TR-1023, CSE Department, University of New South Wales, Australia (December 2010), ftp://ftp.cse.unsw.edu.au/pub/doc/papers/UNSW/1023.pdf
16. Rana, R.K., Chou, C.T., Kanhere, S.S., Bulusu, N., Hu, W.: Ear-phone: an end-to-end participatory urban noise mapping system. In: IPSN 2010: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, pp. 105–116. ACM, New York (2010)
17. Sadler, C.M., Martonosi, M.: Data compression algorithms for energy-constrained devices in delay tolerant networks. In: SenSys 2006: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, pp. 265–278. ACM, New York (2006)
18. Vapnik, V.N.: The nature of statistical learning theory. Springer-Verlag New York, Inc., New York (1995), http://portal.acm.org/citation.cfm?id=211359
19. Xu, Y., Lee, W.C.: On localized prediction for power efficient object tracking in sensor networks. In: ICDCSW 2003: Proceedings of the 23rd International Conference on Distributed Computing Systems, p. 434. IEEE Computer Society, Washington (2003)
20. Xu, Y., Lee, W.C.: Dttc: Delay-tolerant trajectory compression for object tracking sensor networks. In: Proceedings of the IEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2006), pp. 436–445 (2006)
21. Xu, Y., Winter, J., Lee, W.C.: Dual prediction-based reporting for object tracking sensor networks. In: MobiQuitous, pp. 154–163 (2004)

On the Accuracy of Software-Based Energy Estimation Techniques

Philipp Hurni[1], Benjamin Nyffenegger[1],
Torsten Braun[1], and Anton Hergenroeder[2]

[1] Institute of Computer Science and Applied Mathematics (IAM),
University of Bern, Switzerland
{hurni,nyffeneg,braun}@iam.unibe.ch
[2] Institute of Telematics (ITM), Karlsruhe Institute of Technology (KIT), Germany
hergenroeder@kit.edu

Abstract. This paper examines the accuracy of software-based on-line energy estimation techniques. It evaluates today's most widespread energy estimation model in order to investigate whether the current methodology of pure software-based energy estimation running on a sensor node itself can indeed reliably and accurately determine its energy consumption - independent of the particular node instance, the traffic load the node is exposed to, or the MAC protocol the node is running. The paper enhances today's widely used energy estimation model by integrating radio transceiver switches into the model, and proposes a methodology to find the optimal estimation model parameters. It proves by statistical validation with experimental data that the proposed model enhancement and parameter calibration methodology significantly increases the estimation accuracy.

1 Introduction

With energy efficiency being a major concern in the design of Wireless Sensor Networks (WSNs), researchers have thoroughly investigated how to save energy by intelligent design of the communication protocols. On the MAC level, Energy-Efficient Medium Access Control (E^2-MAC) protocols have been proposed to minimize the energy wastage of the radio transceiver, which is typically the major energy consumer of the node onboard components. Most simulation-based E^2-MAC protocol studies rely upon simple energy models of the wireless transceiver chips, with the node's energy consumption being computed as the sum of the energy it spends in the different transceiver states. Most of todays' simulation models implemented in mainstream network simulator frameworks (e.g. ns-2, OMNeT++) distinguish three or even four states (*receive/idle*, *transmit*, *sleep*), as well as switching states with corresponding transition delays.

With research on WSNs becoming more mature, many E^2-MAC protocols have also been prototyped and evaluated on real sensor hardware testbeds. Not surprisingly, experimental validation of E^2-MAC protocols have proven to be much more resource intensive than using mainstream network simulators.

While commonly used networking metrics such as packet delivery rate, source-to-sink latencies or maximum throughput can easily be determined in real-world testbeds, measuring the power consumption of sensor nodes is much harder: costly high-resolution digital multimeters or cathode-ray oscilloscopes need to be hooked to the nodes in order to sample the varying low currents and voltages.

Researchers have henceforth ported the same simple state-based energy estimation models of WSN simulators into their real-world sensor MAC protocols or radio chip drivers. Software-based energy estimation has been proposed in [6] as a viable alternative to using costly hardware-based energy measurement equipment, and has been integrated into the Contiki OS [5] - one of todays' most widespread sensor node operating systems. The Contiki mechanism consists in bookkeeping the time the radio resides in the different transceiver modes on the node itself, and multiplying these times with previously determined power levels to obtain rough estimates for the consumed energy. Many prominent E^2-MAC protocol studies (e.g. [19] [14]) have entirely relied their experimental research results upon the same software-based approach for estimating the energy consumption of their protocol prototypes. More and more recent research papers have utilized exactly this approach (e.g. [8], [2]), although, as already pointed out in [6], no existing study has yet validated the accuracy of this approach with physical hardware-based energy measurements. This paper bridges this missing gap and thoroughly examines the accuracy and the limits of software-based energy estimation on the MSB430 sensor nodes platform [1]. It evaluates several energy estimation models with prototype implementations of 802.11-like CSMA and three E^2-MAC protocols (S-MAC [19], T-MAC [17], WiseMAC [7]). We ran a plethora of experiments under different traffic load levels and with different node instances, in order to statistically describe the achieved estimation accuracies.

The paper is organized as follows: we elaborate on related work on software-based energy estimation and measurement in Sect. 2. In Sect. 3 we introduce the experiment setup for evaluating the different energy estimation models, model enhancements and calibration techniques. Section 4 discusses the observed deviations between different sensor nodes' current draws and their effect on the resulting estimation accuracy. Section 5 evaluates the maximum achievable accuracy of the most widely used energy estimation model (henceforth referred-to as the *Three States Model*) with various wireless channel MAC protocols and traffic rates. We then refine and enhance the estimation model and calibration methodology and experimentally validate the gain in accuracy. Section 6 discusses the maximum accuracy gain that can be achieved with sophisticated and fine-grained parameter calibration. Section 7 concludes the paper.

2 Related Work: Hardware-Based Energy Measurement vs. Software-Based Energy Estimation

In numerous E^2-MAC protocol studies [14] [3], cathode-ray oscilloscopes have been used to quantify the energy consumed by a sensor node in a real-world experiment. The basic idea of the methodology is to connect the sensor node in series with a low-impedance shunt resistor and to measure the resistive voltage drop across the shunt, in order to infer the current flowing through the circuit. This methodology has been applied by a number of studies and can be seen as the *cleanest* approach of energy measurement, as it does not incur any *side-effects* to the sensor node hardware or software. Its main drawback, however, is the costly measurement equipment required and the time-consuming operation of it. Furthermore, if current traces need to be stored in a reasonable resolution during an experiment of several minutes or even hours, the collected raw current traces become huge and quickly cause storage- and memory problems. Only few testbeds have integrated support for distributed real-time energy-measurements, as e.g. MoteLab [18] with some of its nodes, or PowerBench [10]. Hence, in most studies on energy-efficiency issues on the MAC and/or routing layer, researchers have only measured a node's current over a short period of time in order to calibrate a simulation and/or estimation model, and have omitted the energy aspect for the rest of the empirical evaluation.

Sensor Node Management Devices (SNMD) [12] have been developed as a cost-effective alternative to using high-frequency multimeters or oscilloscopes for *side-effect free* high-resolution energy measurement of sensor nodes. SNMDs continuously measure the sensor node current and voltage with resolutions of up to 2 kHz, and therefore need to be connected via USB to a backbone network. This is usually possible in wired stationary testbeds and lab environments, but less in outdoor deployments. With costs of the circuitry components still in the range of 300$, it is a convenient measurement tool for lab environments, but still too costly for large deployments of WSNs or WSN testbeds.

Dunkel et al. [6] motivate the need for software-based *on-line* energy estimation, because only *on-line* estimation mechanisms running on the node itself enable the node to take energy-aware decisions about routing, clustering or transmission power scheduling. The authors experimentally correlate the estimated energy with the sensor nodes lifetime, however underline that "further study is needed to accurately quantify the error rate of the mechanism".

PowerBench [10] partly tackles the issue of the accuracy of software-based energy estimation. The authors elaborate on the difference between their software-based energy estimations (calculated with the commonly used *Three States Model*) and the physically measured energy consumption of the nodes. When running B-MAC [14] and Crankshaft [9], this difference reaches up to 21% of the measurement values. *Per-node-calibration* is shown to vastly reduce this estimation error. With the deviations between software-based estimation and physical measurements still ranging from 2% to almost 14% for some of the examined E^2-MAC protocols, the software-based estimation approach still leaves room for further improvements. The authors further note that frequency of state transitions have a significant impact on the estimation accuracy.

Software-based energy estimation techniques clearly have their advantages and drawbacks. A purely software-based approach can only deliver estimates. It further introduces inherent *side-effects*, as the estimation mechanism itself causes computational costs, which are hard to account for. The advantages, however, are manifold: with an energy estimation being present on the node at run-time, many power-aware WSN algorithms can be applied in real-world deployments. With the WSN field moving from simulation-based towards real-world testbed-based research, finding a simple and painless, but yet accurate methodology for quick and reliable energy estimation can be a significant milestone.

3 Experiment Equipment and Setup

3.1 Sensor Network Management Devices (SNMD)

We used Sensor Node Management Devices (SNMD) [11] to measure and retrieve the node's current and voltage in high resolution, in order to be able to calculate the *physically measured* energy consumption and compare it to software-based estimations later on. SNMDs have been specifically designed to accurately measure current and voltage of sensor nodes with a sampling resolution of up to 20 kHz (up to 500 kHz buffered). SNMDs measure the resistive voltage drop across a 1 Ω shunt resistor. The accuracy of the SNMD has been evaluated using high-precision laboratory equipment for different current ranges. The SNMD firmware corrects each sampled measurement by an error term, which was obtained during evaluative testing in advance. This has been shown to reduce the measurement error introduced by the measurement circuit below $\pm$ 0.5% for any current in the range of 0-100 mA in [12]. As the accuracy of the SNMD has been calibrated using highly accurate state-of-the art measurement equipment, we can safely assume that it provides best possible physical hardware-based energy measurements. Throughout the experimental analysis of this paper, we decided to stick to a sampling rate of 1000 Hz, as the accuracy gain with even higher rates proved to be negligible with the chosen node type and bandwidth settings. Other node platforms, e.g. nodes with IEEE 802.15.4-based radios with higher bandwidth could however probably profit from the high maximum sampling rate of the SNMD.

3.2 The Modular Sensor Boards (MSB430) Platform

The MSB430 node [1] has a CC1020 [16] byte-level radio transceiver operating in the 804-940 MHz ISM frequency band. In its base configuration, the node features a Sensirion SHT11 temperature and humidity sensor, as well as the Freescale MMA7260Q accelerometer. Besides ScatterWeb^2 OS [15], the MSB430 can be run with the popular Contiki OS [5] since recently (v.2.4). While the maximum raw bit rate of the CC1020 is 153.6 kbit/s, the ScatterWeb^2 OS we utilized throughout this paper currently only supports a data rate of 19.2 kbit/s.

3.3 Experiment Setup

We kept the measurement setup as simple as possible, in order to be able to repeatedly perform a significant number of experiment runs with different wireless

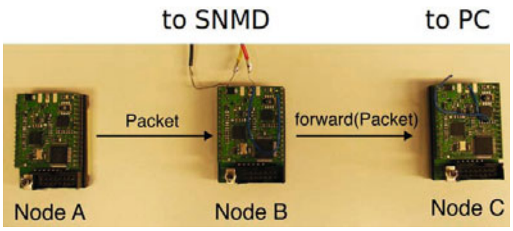


Fig. 1. Node A generating packets, node B hooked to SNMD

channel protocols and traffic rates on the same experiment setup. We lay out nodes A, B, C with a distance of 30 cm on a table, as depicted in Fig. 1. As we wanted to simultaneously obtain both the software-based estimations and the unaffected physical hardware-based measurements of the same node B, we had to keep node B unplugged from any serial interface, as the node would otherwise draw some small current from the powered USB serial interface cable. Hence, in order to obtain the software-based estimations of node B without accessing it over a serial cable, we let node B write its energy estimation model data (time in transmit mode, time in receive mode, etc.) into the packet payload.

Packets are 50 bytes each (10 bytes header, 40 bytes payload). In each experiment run, node A starts sending constant-rate traffic of rate r towards node B during $T_{exp} = 600s$. Right after the reception of the first packet, Node B starts keeping track of the time its transceiver resides in the different states. After injecting its estimation model data into the packet, Node B forwards the packets to node C, which decapsulates the packet and logs node B's energy estimation data to the serial interface, which is connected to a Desktop PC. During the entire experiment, the current trace of node B is read from the SNMD's serial interface, which is connected to the same Desktop PC. As discussed later in the analysis, we varied the traffic rate r at node A from very low rates (1 packet every 100s) to high rates (2 packets/s) with each different wireless channel MAC protocol. We measured 10 independent runs for each setting, and evaluated different node instances. In Sect. 4, node B (the *measurement node*) was exchanged with other node instances of the same type.

4 Hardware-Dependent Energy Consumption Deviations

Applying software-based energy *estimation* inevitably introduces *inaccuracies*. The differences between the *estimated* power consumption and the *physically measured* power consumption can generally be explained by the slightly differing behavior of the nodes' electronic hardware components, or may stem from the inherent imperfection of the software-based model and the applied estimation methodology. This section elaborates on the effect of the slight deviations on the power consumption of different node instances of the same node type, whereas Sect. 5 discusses the impact of choosing an appropriate estimation model and calibrating the model parameters.

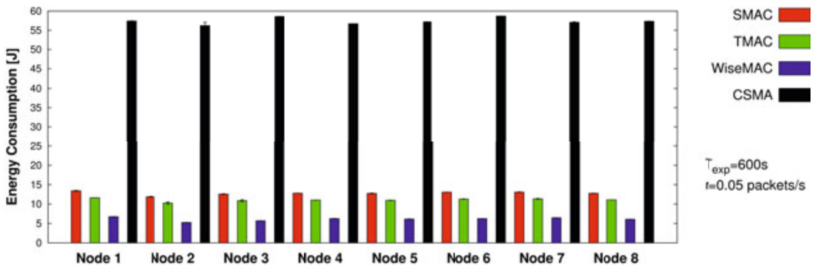


Fig. 2. Energy consumed by 8 different instances of nodes

4.1 Different Current Draws with Different Nodes

As discovered in previous experimental studies [13] [10], the power consumption of different instances of the same type of sensor node often varies in the range of some few percent. [13] presumes that this variation stems from differences in the electronic components tolerances. We hence first examined multiple instances of MSB430 nodes running different wireless MAC protocols, given a constant traffic rate of 1 packet each 20s over $T_{exp} = 600s$. With this evaluation, we quantify the estimation inaccuracies caused by the variation in the energy consumption of different instances of the same node type - in our case the MSB430 platform.

Figure 2 depicts the energy consumed by eight different instances of MSB430 nodes and the four examined protocols during 10 experiment runs. Each bar depicts the mean value and standard deviation measured during 10 independent runs - the latter was low in most cases and is hence barely visible. The energy consumption obviously varies heavily from protocol to protocol (eg. WiseMAC vs. CSMA). The variation from node to node however is also clearly visible, e.g. the energy consumed by node 6 running CSMA is roughly 4% higher than that of node 2. We investigated the reason for these differences in the current traces and found that indeed, the current drawn from different nodes can vary to a certain degree, and that the variation can even differ for each of the different transceiver states. Figure 3 depicts the current traces of nodes 1 and 2 running CSMA and receiving a data packet, and sending it further to another node. As one can clearly see, node 1 draws approx. 2 mA less than node 2 when transmitting. Although the transmission power settings were set identically for all nodes, the

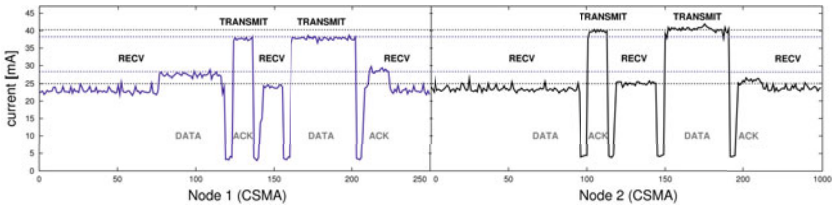


Fig. 3. Current draw of nodes B and C

current levels in the transmit state obviously varied to a certain degree. A further anomaly we encountered is that some nodes drew more current in receive mode when actually receiving data compared to listening to an idle channel, whereas in most cases, no significant difference between these two cases could be measured. This effect is visible in Fig. 3 as well: node 1 consumes approximately 3 mA more when receiving data, compared to node 2 which consumes more or less the same current when receiving data or listening to an idle channel. As both nodes are running the same interrupt service routine code and did not run any other computationally intensive tasks during this time, the CPU can neither be held accountable for this effect. We further discovered slight differences in the peak energy consumption as well as in the duration of transceiver switches depending on the protocol, and even depending on the traffic load. We presume that these differences stem from the inaccuracies in the production of the electronic components. Fast switching between the different operation modes of the radio could probably also have a temporary impact on the behavior of active circuit elements. Although the temperature is known to impact on the power consumption of electronic devices, we can safely exclude this as an explanation for the discovered deviations, as all experiments were run under room temperature in the same laboratory environment.

4.2 Statistical Characterization of Node Deviations

In an attempt to quantify the discovered differences between the eight measured node instances, we a) determined the mean and standard deviation of the measured energy consumptions of all measurement runs of all eight nodes for the CSMA protocol in the experiment described in 3.3 (with $T_{exp} = 600s$ and a traffic rate r of 0.05 packets/s), and b) compared each pair of nodes to determine the the maximally differing nodes. We chose CSMA because at examined traffic rates, no packet loss occurred within all CSMA runs. Hence, the CSMA experiment runs were most suited for examining the per-node differences.

a) The mean consumed energy of the eight different nodes throughout T_{exp} was 57.55 Joules with a standard deviation of 1.54%. Hence, roughly two thirds of all node instances exhibit a value in-between 57.55 Joules $\pm$ 1.54%, given that the variation between different nodes follows normal distribution. We conjectured that the latter is the case, as Jarque-Bera's test on the normality of the measurement variation (JB-value: 0.701) could not be rejected (cf. [4]).

b) The maximum deviation between the mean measured energy consumption of the two maximally differing nodes was determined to be 4.24% (of the respective higher value). We tested the claim that these two nodes do actually differ significantly from each other, i.e. that the discovered deviations are not caused by coincidence or the limited set of observations. We found that the null-hypothesis of a two-sided t-test claiming that the two nodes exhibit the same mean energy consumption (=on average consume the same amount of energy) could safely be rejected at the 95% confidence level. This however was not the case for all the node pairs, as some groups of nodes obviously exhibit similar patterns in their energy consumption (cf. Fig. 2).

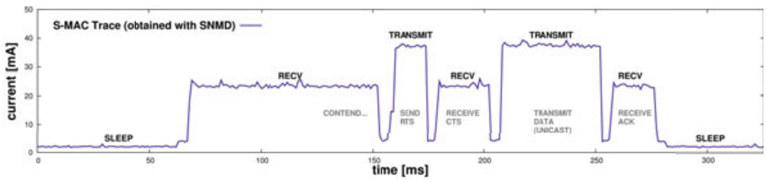


Fig. 4. Current Draw of node B

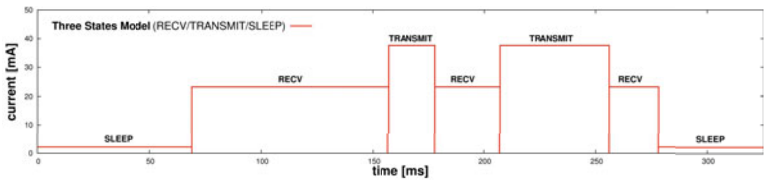


Fig. 5. Node B as modeled by the *Three States Model*

5 Evaluating Software-Based Energy-Estimation Models

In this section we analyze the impact of the choice of the estimation model on the resulting estimation accuracy for experiments with different traffic load levels and wireless channel protocols. With the variation between different nodes being in the range of more than 4% for the specified experiment scenario, we decided to use exactly *the same sensor node* and also the *same SNMD device* throughout the entire analysis in this section, in order not to introduce variations caused by differing measurement hardware and measured hardware.

5.1 Three States Model (recv/idle, transmit, sleep)

The most frequently used model to date for estimating a node's energy consumption - especially in E^2-MAC protocol studies - consists in modeling the latter as a function of the three states of the radio transceiver *receive/idle listening*, *transmit* and *sleep* (cf. [10] [19] [14]). We henceforth refer to this model as the *Three States Model*. The Contiki OS (v. 2.4) energy estimation mechanism models the radio's power consumption using this model, but *separately* tries to keep track of the CPU power consumption, which can vary depending on the Low-Power-Mode (LPM) it is currently operating. The ScatterWeb^2 OS used in this study puts the CPU to LPM1 as soon all events have been processed, where the node's current is approximately 1.8 mA, given that the radio is turned off. With the CPU active and the radio off, the node current is roughly 3.5 mA. As our examined E^2-MAC protocols generally do not incur intensive computations, we neglected to account for the CPU costs separately, and considered the CPU's power consumption to be *integrated* within the three states of the transceiver. Estimating the CPU power consumption in software when applying E^2-MAC protocols is anyway not easy to achieve, as most of the MAC-related CPU activity takes place in interrupt service routines. Accounting for such may even cause more costs than the protocol-related computations themselves (c.f. [13]). If the CPU activity does not vary much across state changes of the radio transceiver, modeling the CPU and radio integrally safely holds. Figure 4 illustrates that for the given E^2-MAC protocol, accounting for CPU in a combined manner with the three different power levels of the radio transceiver is sufficient.

We henceforth modeled the energy consumption of our S-MAC [19], T-MAC [17], WiseMAC [7] and CSMA implementations using the abovementioned *Three States Model.* We let the nodes keep track of the time differences between the transceiver switches, in order to determine how much time has been spent in each state. Figure 4 depicts the current draw during the active interval of an S-MAC frame containing an RTS/CTS handshake and a subsequent data packet transmission. Figure 5 illustrates how this current draw is being approximated by the *Three States Model.* The total energy consumed (denoted as E) corresponds to the area below the current draw multiplied by the supply voltage, which is assumed to be constant. Analytically, the *Three States Model* can be formulated as equation MI. The consumed energy E is calculated as the sum of the total time spent in the receive state multiplied by the respective power level $T_{rcv}P_{rcv}$, and the respective terms for the transmit and sleep states ($T_{slp}P_{slp}$ and $T_{tx}P_{tx}$). This approach is identical to the one applied in [10], [19] and [14].

$$E = P_{rcv}T_{rcv} + P_{tx}T_{tx} + P_{slp}T_{slp} = I_{rcv}V_{rcv}T_{rcv} + I_{tx}V_{tx}T_{tx} + I_{slp}V_{slp}T_{slp} \quad \text{(MI)}$$

Parameter Definition through Example Measurement: [19], [14], [3] and [10] calibrate the parameters of their energy model by measuring the currents the nodes draw in the different states, and multiplying it with the supply voltage to obtain P_{rcv}, P_{tx} and P_{slp}. They do so by using either oscilloscopes or high-precision multimeters and by measuring the current in each state over a

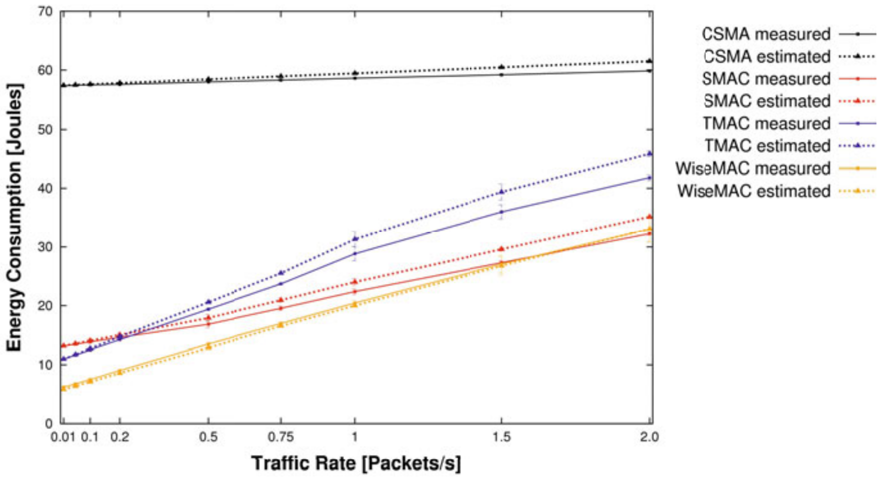


Fig. 6. Measured vs. Estimated Energy Consumption

certain timespan. In the first attempt, we pursued exactly the same approach, and determined the mean values of I_{rcv}, I_{tx}, I_{slp} by measuring each state of the *measurement* node using the SNMD for a couple of seconds. The stable mean values were determined to be 23.5353 mA, 37.4872 mA and 2.1495 mA for I_{rcv}, I_{tx}, I_{slp}, respectively. We further set the voltage according to the supply voltage of the SNMD to $V_{rcv} = V_{tx} = V_{slp} = 4.064V$.

Figure 6 depicts the mean values of the energy measurements and the estimations being computed with the *Three States Model* - using the parameters for P_{rcv}, P_{tx} P_{slp} measured in the example trace. One can clearly see that the estimations fit quite well for low traffic rates, but that the gaps between mean estimations and mean measurements become larger with higher rates of packets being sent over the *measurement* node. For most protocols - especially S-MAC and T-MAC - the energy estimation over-estimates the energy consumed by the node with increasing load. This increasing over-estimation stems from the fact that the *Three States Model* does not account for the transceiver switches. As one can clearly see comparing Fig. 4 with Fig. 5, the current draw decreases to roughly 4 mA when the transceiver is switched to receive or transmit - hence drawing less current than estimated with the *Three States Model.* By defining parameters through example measurement, the impact of the applied traffic load and the frequent transceiver switches as well as the particularities of the MAC protocol are not being taken into account at all. Extrapolating from a short example measurement of a node hence leads to suboptimal parameters for the *Three States Model*, even when using the same node for parameter calibration and the evaluation of the accuracy.

Parameter Definition through Ordinary Least Squares (OLS): Being able to physically measure the current draw of a sensor node *and* at the same time obtain the software-based estimation calculated by the node itself offers the opportunity to relate the estimations to the real-world measurements. Using the plethora of experimental data gained in the many experiments runs (in total over 12 GB), we reflected upon a method to determine more resilient parameters for the unknown variables P_{rcv}, P_{tx}, P_{slp} of the *Three States Model.* Ideally, the software-based energy estimation running on the node should neither rely on the particularities of a specific MAC protocol, nor on the shape or intensity of the traffic. *Ordinary Least Squares (OLS) Regression Analysis* yielded the most suitable technique to determine the unknown variables for a linear estimation model with multiple unknown variables. OLS minimizes the sum of squared errors (SSE) between estimations and observations (= the measurements). We formulated a multivariate OLS regression model with the *explanatory variables* T_{rcv}, T_{tx}, T_{slp} (the times spent in the different transceiver states, calculated at runtime), as well as the physically measured *dependent variable* E obtained using the SNMD device. The resulting estimation equation hence simply comprises equation MI and the error term ε for the residuals.

$$E = P_{rcv}T_{rcv} + P_{tx}T_{tx} + P_{slp}T_{slp} + \varepsilon \qquad \text{(OLS-I)}$$

With the above multivariate OLS model, the unknown parameters are estimated as the OLS estimator $\hat{\beta} = (\hat{P_{rcv}}, \hat{P_{tx}}, \hat{P_{slp}})$ which calculates as

$$\hat{\beta} = ((X'X)^{-1}X')y$$

where X is the matrix of all the observations of the *explanatory variables*, consisting in 3 columns (T_{rcv}, T_{tx}, T_{slp}) and a row for each measurement, and y the vector with the corresponding observations of the *dependent variable.* We assessed the coefficient of determination R^2 to measure the *goodness of fit* of the multivariate linear regression model and obtained a surprisingly high value of $R^2 = 0.9980$.

Estimation Accuracy of the Three States Model: In order to determine the accuracy of the OLS-calibrated software-based model, a cross-validation with totally new experimental data is inevitable to omit overfitting effects (cf. [4]). The determination of the parameters P_{rcv}, P_{tx}, P_{slp} using OLS regression was hence achieved on a first set of experiment runs, the so-called *training set.* The results concerning the estimation accuracy of this section however were gained with a new set of experimental data, to which we will further refer as *validation set.* We fed $\hat{\beta}$ containing the OLS estimators of the unknown variables $\hat{P_{rcv}}, \hat{P_{tx}}, \hat{P_{slp}}$ into the node's estimation model and estimated the energy consumption with the validation set. We considered the so-called *mean absolute error (MAE)* (=the average difference, cf. [4]) between the estimations and the measured values to be the best statistical measure for the *accuracy* of the employed *Three States Model.* The MAE and its standard deviations calculated across all protocols and traffic rates in the validation set (henceforth always given as percentage of the SNMD-measured values) is depicted in Fig. 7. For each traffic rate, the estimation error using the OLS estimator parameters is 4.2% to 35.9% lower than the corresponding error when using the model parameters defined through example measurement. Across all measurements, the mean absolute estimation

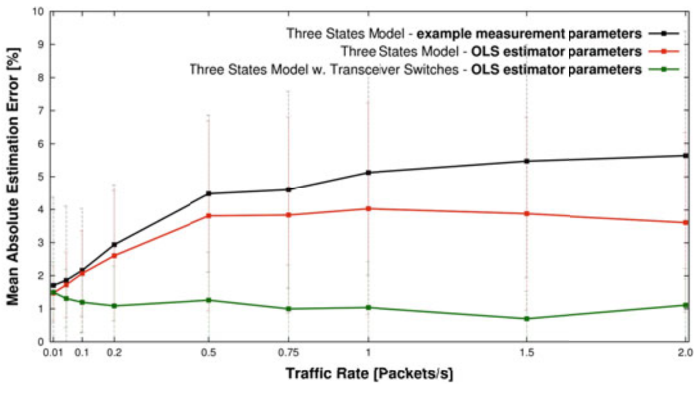


Fig. 7. Absolute Mean Estimation Error (in %) vs. Traffic Rate (packets/s)

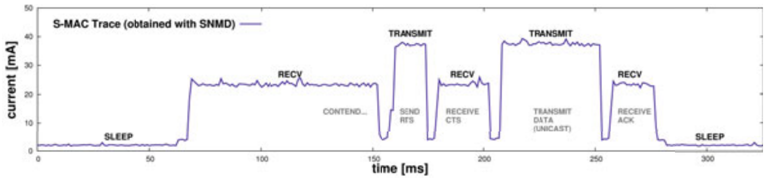


Fig. 8. Current Draw of node B

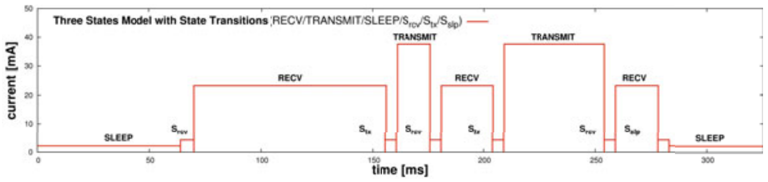


Fig. 9. Current modeled by the *Three States Model with State Transitions*

error and standard deviation (denoted as $\mu \pm \sigma$) of the *Three States Model* with the parameters defined by example measurement equals 3.77% $\pm$ 3.17%. When determining the parameters by OLS, we obtain 3.00 % $\pm$ 2.55% - hence achieving an overall MAE reduction error by 21%.

5.2 Three States Model with State Transitions

With the mean absolute estimation error still in the range of 3% or more, we investigated further means to improve the estimation accuracy. As Fig. 8 exhibits, the current draw temporarily drops to approximately 4 mA during the state switches. These state switches remain unaccounted for in the OLS regression model specified in equation OLS-I. We first attempted to sum up the transition times between the transceiver states. This approach however led to unsatisfactory results, as the ScatterWeb[2] OS only supports a clock in milliseconds precision. Simply counting the transceiver switches and integrating them into the OLS regression model however led to a significant improvement in the estimation accuracy. The number of transceiver switches (*from* an arbitrary state) *to* the *receive*, *transmit* or *sleep* state was hence accounted for with the additional regressands s_{rcv}, s_{tx}, and s_{slp}. We refer to this model as the *Three States Model with State Transitions* hereafter, as specified in equation MII.

$$E = T_{rcv}P_{rcv} + T_{tx}P_{tx} + T_{slp}P_{slp} + \alpha s_{rcv} + \beta s_{tx} + \gamma s_{slp} \qquad \text{(MII)}$$

According to this enhanced model, the energy consumed by an arbitrary node is a function of the total time it has its radio transceiver in the three different states (denoted as T_{rcv}, T_{tx}, T_{slp}) and the three adjustment terms αs_{rcv}, βs_{tx}, and γs_{slp}. The parameters α, β, γ compensate for the transceiver switches to the states *receive*, *transmit* and *sleep*. Their optimal values are determined empirically using OLS regression.

Parameter Definition through Ordinary Least Squares (OLS): We specified the corresponding OLS regression model to equation MII with the *explanatory variables* T_{rcv}, T_{tx}, $T_{slp}, s_{rcv}, s_{tx}, s_{slp}$, as well as the *dependent variable* E (for which we obtain the *real measured value* using the SNMD device) as

$$E = P_{rcv}T_{rcv} + P_{tx}T_{tx} + P_{slp}T_{slp} + \alpha s_{rcv} + \beta s_{tx} + \gamma s_{slp} + \varepsilon \qquad \text{(OLS-II)}$$

The OLS estimator $\hat{\beta} = (\hat{P_{rcv}}, \hat{P_{tx}}, \hat{P_{slp}}, \hat{\alpha}, \hat{\beta}, \hat{\gamma})$ is calculated in analogy to Sect. 5.1. We obtained a coefficient of determination of $R^2 = 0.9998$ for the multivariate linear regression model OLS-II, a slightly higher value than for OLS-I. However, when comparing the *goodness of fit* of two regression models, the R^2 indicator is not a meaningful criterion, as it never decreases when adding more regressands. The so-called *adjusted coefficient of determination* $\bar{R}^2$ (cf. [4]) adjusts for the number of explanatory terms in a model. Unlike R^2, this coefficient only increases when the increase of explanatory variables actually improves the model. An increase of $\bar{R}^2$ upon addition of an explanatory variable to a multivariate OLS model is hence generally understood as a proof that the new model delivers a better fit to the measured data. An even better coefficient for comparing the *goodness of fit* of two regression models however is the Akaike Information Criterion (*AIC*) (cf. [4]). The lower the *AIC* value, the better the fit to the model. We measured the $\bar{R}^2$ and *AIC* coefficients before and after adding the transceiver switches s_{rcv}, s_{tx}, s_{slp}, to the OLS model (OLS-I vs OLS-II). With $\bar{R}^2$ increasing from $\bar{R}^2_I = 0.9801$ to $\bar{R}^2_{II} = 0.9980$, and *AIC* decreasing from $AIC_I = 2.5036$ to $AIC_{II} = 0.2154$, we can safely claim that the *Three States Model with State Transitions* delivers a significantly better fit to the measurement data than the today's most widely used simple *Three States Model.*

Estimation Accuracy of the Three States Model with State Transitions: We calibrated the OLS estimators for the parameters of the second model with the *training set*, and examined the resulting estimation accuracy on the *validation set*. Across all measurements, the MAE and standard deviation (denoted as $\mu \pm \sigma$) of the software-based estimations using the *Three States Model with State Transitions* (and the parameters determined by OLS) compared to the physically measured values equals 1.13% ± 1.15%. Comparing this result to the 3.00 % ± 2.55% obtained with the *Three States Model* (and the parameters determined by OLS), our proposed model enhancement led to an overall reduction of the MAE by remarkable 62.3% (cf. Fig. 7).

6 The Impact of Calibration on the Estimation Accuracy

This section evaluates the impact of different possible granularities of *calibration* on the achievable accuracy of the software-based energy estimation technique. Throughout this section we henceforth utilize the same multivariate OLS regression methodology and the *Three States Model with State Transitions* as described in Sect. 5, as applying this model generally led to the lowest estimation errors.

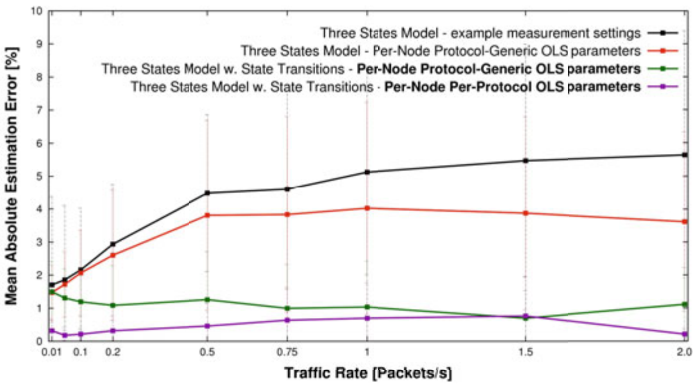


Fig. 10. Absolute Mean Estimation Error (in %) vs. Traffic Rate (packets/s)

6.1 Per-Node Calibration

Different wireless sensor node instances often exhibit a slightly different behavior with respect to their power consumption levels in the different transceiver states. This effect has been observed in previous studies [13] [10], and has been quantified for the utilized MSB430 platform in Sect. 4. We have encountered node pairs of the same node type that differed by more than 4% in their physically measured energy consumption. Hence, even the best *node-generic* software-based energy estimation mechanism can be more than 4%, if its underlying model parameters were not calibrated on a *per-node* basis.

Researchers intending to calibrate their energy estimation model with only one particular sensor node instance must therefore be aware that their energy consumption estimates will deviate from the *real* energy consumption by the unavoidable hardware-based variation, unless each node has previously been calibrated individually. Calibrating on a *per-node* basis however means that *every single node* needs to be physically measured (e.g. with an SNMD or a high-resolution multimeter) ideally with different MAC protocols and different traffic rates. Only this time-intensive calibration leads to the set of *per-node* but *protocol-generic* estimation model parameters which has been shown in Sect. 5.2 to reduce the mean absolute estimation error ($\mu \pm \sigma$) to 1.13% $\pm$ 1.15%.

6.2 Per-Node and Per-Protocol Calibration

In Sect. 5.2, we intentionally *generalized* from the particularities of the MAC protocol by running OLS over four different MAC protocols. Hence, we obtained protocol-independent (but node-specific) estimation parameters. In order to obtain *per-protocol* (and node-specific) calibrated OLS estimator parameter values, the methodology applied in Sect. 5 can be applied without any adaptation. However only the observations of the specific protocol and node have to be chosen from the training set in order to calculate the OLS estimator. The same *specialization* effect can also be achieved by supplying more information to the OLS regression model with introducing so-called *dummy variables* that indicate the currently used protocol (cf. [4], p. 299ff). We propose this *per-protocol* calibration as an even more accurate estimation approach, which might be useful if researchers know exactly what protocol they intend to use on the MAC layer in advance. We calculated different OLS parameter sets for each of the four protocols (S-MAC, T-MAC, WiseMAC, CSMA) and used the same node (node 1 in Fig. 2) used in Sect. 5 for calculating the resulting accuracy on the validation set. The combined approach of *per-node and per-protocol* calibration obviously leads to the highest accuracy. Across all four protocols and traffic rates, we obtained a mean estimation error and standard deviation ($\mu \pm \sigma$) of only 0.42% ± 0.72%. The combined calibration approach however has multiplicative impact on the overhead before network deployment, as all nodes need to be equipped with tailor-made estimation model parameters for each protocol. Figure 10 illustrates the different estimation errors measured when applying the the *per-node and protocol-generic* or the *per-node and per-protocol* calibration approach.

7 Conclusions

This paper evaluates the accuracy of software-based energy estimation models on the MSB430 platform. We have identified and quantified the different factors which cause deviations of the software-based estimations from the *real* physically measurable energy consumption. The inaccuracies in the production of the electronic components have been shown to impact on different power consumption levels, which led to nodes differing by more than 4% in their energy consumption. The paper conveys that software-based energy estimation can be a valuable alternative to using sophisticated measurement hardware, especially in outdoor-deployments where the latter is impossible - at least for evaluating protocols where the CPU is used frugally, i.e., E^2-MAC or routing protocols. Enhancing today's most widely used simple *Three States Model* with information regarding the state transitions and applying multivariate OLS regression to calibrate the model parameters has been shown to remarkably reduce the estimation error. The mean absolute error (MAE) and standard deviation ($\mu \pm \sigma$) of the energy estimations of the software-based model using protocol-generic but per-node calibrated parameters could be pushed to as few as 1.13% ± 1.15%. Applying even more sophisticated parameter calibration of per-node *and* per-protocol calibration has been shown to reduce the mean absolute error and standard deviation to as few as only 0.42% ± 0.72% across the four evaluated wireless channel MAC protocols S-MAC, T-MAC, WiseMAC, and 802.11-like CSMA.

References

1. Baar, M., Koeppe, E., Liers, A., Schiller, J.: The ScatterWeb MSB-430 Platform for Wireless Sensor Networks. In: SICS Contiki Workshop (2007)
2. Boano, C.A., Voigt, T., Tsiftes, N., Mottola, L., Römer, K., Zúñiga, M.A.: Making Sensornet MAC Protocols Robust against Interference. In: Silva, J.S., Krishnamachari, B., Boavida, F. (eds.) EWSN 2010. LNCS, vol. 5970, pp. 272–288. Springer, Heidelberg (2010)

3. 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, pp. 307–320 (2006)
4. Draper, N., Smith, H.: Applied Regression Analysis. Wiley Series in Probability and Statistics. Wiley-Interscience, Hoboken (1998)
5. Dunkels, A., Grönvall, B., Voigt, T.: Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. In: IEEE EmNets, pp. 455–462 (2004), `http://www.sics.se/contiki/`
6. Dunkels, A., Osterlind, F., Tsiftes, N., He, Z.: Software-based On-line Energy Estimation for Sensor Nodes. In: IEEE EmNets, pp. 28–32 (2007)
7. El-Hoiydi, A., Decotignie, J.-D.: WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks. In: Nikoletseas, S.E., Rolim, J.D.P. (eds.) ALGOSENSORS 2004. LNCS, vol. 3121, pp. 18–31. Springer, Heidelberg (2004)
8. Finne, N., Eriksson, J., Tsiftes, N., Dunkels, A., Voigt, T.: Improving Sensornet Performance by Separating System Configuration from System Logic. In: Silva, J.S., Krishnamachari, B., Boavida, F. (eds.) EWSN 2010. LNCS, vol. 5970, pp. 194–209. Springer, Heidelberg (2010)
9. Halkes, G., Langendoen, K.: Crankshaft: An Energy-Efficient MAC-Protocol for Dense Wireless Sensor Networks. In: Langendoen, K.G., Voigt, T. (eds.) EWSN 2007. LNCS, vol. 4373, pp. 228–244. Springer, Heidelberg (2007)
10. Haratcherev, I., Halkes, G., Parker, T., Visser, O., Langendoen, K.: PowerBench: a Scalable Testbed Infrastructure for Benchmarking Power Consumption. In: International Workshop on Sensor Network Engineering (IWSNE) (2008)
11. Hergenröder, A., Horneber, J., Meier, D., Armbruster, P., Zitterbart, M.: Distributed Energy Measurements in Wireless Sensor Networks. In: ACM SenSys, Demo Session (2009)
12. Hergenröder, A., Wilke, J., Meier, D.: Distributed Energy Measurements in WSN Testbeds with a Sensor Node Management Device (SNMD). In: Müller-Schloer, C., Karl, W., Yehia, S. (eds.) ARCS 2010. LNCS, vol. 5974. Springer, Heidelberg (2010)
13. Landsiedel, O., Wehrle, K., Goetz, S.: Accurate Prediction of Power Consumption in Sensor Networks. In: IEEE EmNets (2005)
14. Polastre, J., Hill, J., Culler, D.: Versatile Low Power Media Access for Wireless Sensor Networks. In: ACM SenSys, pp. 95–107 (2004)
15. ScatterWeb^2 OS - Freie Universität Berlin & ScatterWeb GmbH: `http://scatterweb.mi.fu-berlin.de/` and `http://www.scatterweb.de`
16. Texas Instruments CC1020: Single-Chip FSK/OOK CMOS RF Transceiver
17. Van Dam, T., Langendoen, K.: An Adaptive Energy Efficient MAC Protocol for Wireless Sensor Networks (TMAC). In: ACM SenSys, pp. 171–180 (2003)
18. Werner-Allen, G., Swieskowski, P., Welsh, M.: MoteLab: a Wireless Sensor Network Testbed. In: Information Processing in Sensor Networks (IPSN) (2005)
19. Ye, W., Heidemann, J., Estrin, D.: An Energy Efficient MAC Protocol for Wireless Sensor Networks. In: INFOCOM, pp. 1567–1576 (2002)

Fast, Accurate Event Classification on Resource-Lean Embedded Sensors

Hao Jiang and Jason O. Hallstrom

School of Computing, Clemson University
Clemson, SC, USA
{hjiang,jasonoh}@cs.clemson.edu

Abstract. In wireless sensing applications, it is often necessary to identify high-level events based on low-level sensor signals. Due to the limited computing and energy resources available on existing hardware platforms, achieving high precision classification of high-level events *in-network* is a challenge. In this paper, we present a new classification technique for identifying events of interest on resource-lean sensors. The approach introduces an innovative condensed kd-tree data structure to represent processed sensor data and uses a fast nearest neighbor search to determine the likelihood of class membership for incoming samples. The classifier consumes limited resources and provides high precision classification. To evaluate the approach, two case studies are considered, in the contexts of human movement and vehicle navigation, respectively. The classification accuracy is above 85% across the two case studies.

1 Introduction

Wireless sensor networks (WSNs) [1] offer the potential to identify high-level events using simple sensor signals. Event detection involves extracting information from raw sensor readings and reporting the occurrence of interesting events in the physical world. The detection challenge stems from the resource constraints associated with common hardware platforms.

A great deal of work has focused on event detection in sensor network systems, particularly in the context of accelerometer data – also our focus. Due to their relatively inexpensive price, accelerometers are largely available for standard sensor nodes and mobile phones. By observing and analyzing accelerometer readings, rich information regarding movement, tilt, speed, and vibration can be extracted. Consider some of the representative application areas. People-centric event detection systems focus on the analysis of human movement using wearable sensor nodes. A wearable system is often able to detect walking, sitting, standing, and other human behaviors performed by the carrier [6, 8, 10]. This type of system is also applied in clinical research [3, 16], for instance, in studying the movement of Parkinson's patients undergoing particular drug therapies. Others have considered the identification of context information based on sensor movements. Nericell [19] uses a mobile phone accelerometer to identify traffic and road conditions when carried by a driver. Other application areas involve structural vibration monitoring of bridges [11, 12], buildings [26], roads [13], and even volcanos [24]. In these projects, imperceptible vibrations are collected, and events of interest are extracted and recorded.

To detect interesting events, an accelerometer must typically provide a high sampling rate. If a 2-axis accelerometer is used with a 16-bit ADC, 240KB of raw acceleration data is produced if the sensor is sampled for 10 minutes at 100Hz. If a 30 byte packet payload is used, it requires at least 8000 packets to transmit this data, assuming single hop communication and zero packet loss. (In TinyOS [14], the maximum payload size is 28 bytes by default.) This is not a feasible choice given the resource constraints of the target platforms.

As a result, feature extraction and/or data compression techniques are often applied [11, 19, 24]. However, these techniques often rely on time-consuming manual observation and analysis of the characteristics of the data. Further, these techniques often target a narrow set of data (e.g., specific acoustic samples); solutions may not be transferable to other scenarios. Finally, for acceleration-based detection, a fixed node orientation is typically required.

In contrast, as an alternative approach, machine learning and pattern recognition techniques enable automation and transferability. Machine learning techniques are used to generate classification functions based on empirical data collected during a training phase. The generated functions are used to classify incoming sensor data into one or more groups. However, traditional classification techniques are computationally prohibitive for most sensor nodes. One solution is to collect sensor readings and process the data on a PC/server [3, 9, 23], but again, the cost of communication is high. One collateral effect is that the high cost, particularly in terms of energy, inhibits re-training, which is necessary in dynamic environments.

The main challenge addressed by our work centers on the mismatch between computationally intensive classification techniques and resource-constrained sensor nodes. We present a generic, node-level classification technique for resource-constrained sensors, such as the popular Tmote platform, with an MSP430 microprocessor. We also present preprocessing techniques for accelerometer data, designed to enhance classification accuracy. In the classifier design, we use an innovative condensed kd-tree, which can reduce the number of nodes in a standard kd-tree by 90.0%. Adapting the nearest neighbor search to the condensed tree, we classify incoming events in $O(n^{\frac{1}{2}})$ time for the 2-dimensional case, where n denotes the tree size. The preprocessing technique uses a sliding window to smooth the accelerometer readings in $O(1)$ time; this improves the performance of the classifier significantly. The classifier yields high classification accuracy in our case studies and reduces communication overhead and energy consumption compared to the raw data collection approach. Moreover, since the classifier is generated in-network, re-training is energy-efficient.

The main contributions of our work are: (1) We present a new classification technique based on a condensed kd-tree data structure, using a dynamically-scoped nearest neighbor search; (2) We describe preprocessing techniques for accelerometer data to improve classification accuracy; (3) We apply (1) and (2) in the context of two case studies based on prototype implementations for TinyOS. The classification accuracy is above 85% in the two case studies considered.

Paper Organization. Section 2 summarizes the most relevant related work. Section 3 describes the condensed kd-tree design, the fast classification technique, and the preprocessing strategy for accelerometer data. Section 4 describes the system implementation and experimental setup used in our analysis and the two case studies used to evaluate our classification approach. Section 5 summarizes our contributions.

2 Related Work

A number of other authors have investigated event detection using classification-based techniques. We briefly describe some of the work most relevant to ours.

Ganti et al. describe SATIRE [6], a software architecture for wearable sensor networks that includes services for accelerometer sampling, data storage, and data transmission, as well as a web-based data portal. A Hidden Markov Model (HMM) is used to classify human activities and find possible hidden states – unobserved states under the assumption of a Markov process; the processing is performed by an upper-tier host application. A similar approach is seen in the work of He et al. [9]. The authors use the Viterbi algorithm [5] to find the most likely sequence of hidden states in a HMM, and again, the algorithm is applied on an upper-tier host. A sliding window preprocessing scheme that computes the arithmetic mean within each interval is applied to reduce the communication overhead between nodes and the host. The Tmote Invent platform is used as a wearable device in their project, the same type of sensor used in our work. Lorincz et al. describe Mercury [16] to sense abnormal patient activity using a wearable sensor network. Sensor nodes log all collected data to flash storage and transmit a small portion of the collected data back to a host server; five standard features are extracted on the sensor nodes. The authors describe a throttling driver that coordinates data downloads based on configured feature thresholds and a target battery lifetime. Their system does not include in-network machine learning; event detection is delegated to a host server.

Miluzzo et al. present CenceMe [18], a smartphone application designed to detect user-centric events using audio and accelerometer data. A partially *on-phone* classification algorithm is implemented in their work. Classifier training is performed on a desktop machine, and a decision tree is generated using the J48 decision tree algorithm [25]. The generated decision tree is exported to a resource-rich smartphone, which processes raw data using a Discrete Fourier Transform (DFT) and classifies the resulting data. Lu et al. present Sound-Sense [17], an event detection application that classifies daily environmental sounds using a smartphone. This application preprocesses raw sound data and uses coarse classification to classify the resulting data into groups. It then classifies each group into finer "intra-category" subdivisions. Unrecognized sounds are categorized into new classes based on a Mel Frequency Cepstral Coefficient

(MFCC) feature vector [15]. The system is capable of distinguishing a number of common sounds. It is unclear where the training phase is performed.

Mohan et al. present Nericell [19], a system used to monitor road and traffic conditions using GPS, microphone, and accelerometer data from a smartphone. Accelerometer data is used to detect braking events and road bumps. A key contribution is the presentation of a 3-axis accelerometer reorientation algorithm. Machine learning is not used in this paper; the classification results stem from manual analysis of the features of the sensor data. For example, bump detection is achieved by comparing vertical acceleration readings with an acceleration threshold based on traveling speed, derived from empirical observation.

Kim et al. [13] present a classification approach used to detect military vehicles using acoustic and seismic sensors deployed on a road. The Gaussian Mixture Model (GMM) algorithm [21] is applied as a basic classifier, and the resulting likelihood measurements are processed through a decision tree generated using the Classification and Regression Tree (CART) [2] algorithm. The computation is performed on an upper-tier host. Kim et al. [11] present work focused on vibration monitoring using sensors attached to a bridge. High-rate accelerometer sampling and data transmission techniques are used in their approach. Event detection and classification are performed on a resource-rich host.

In contrast to the above work, we describe a complete training and classification approach for resource-constrained sensor nodes. Our work assumes the absence of a basestation.

3 Condensed kd-Tree Classifier Design

In this section, we present the design of the condensed kd-tree classifier. We employ a condensing technique to store the tree using limited memory. The classifier training phase involves processing training data elements labeled with their respective class designations, which are then inserted into the tree; a classifier is generated as the output of this stage. The classification phase uses the classifier to assign incoming data elements to their respective classes.

3.1 kd-Tree Data Structure

A kd-tree is a binary tree data structure which is broadly used to solve geometry problems and can be used as a classifier using k-nearest neighbor search [4, 7, 22]. A kd-tree is built by alternately splitting the point set by half in one of the dimensions of the pattern space. As shown in Fig. 1, a kd-tree node splits the point set evenly into two sets based on the median x coordinate, then y coordinate, and splits the resulting set on x again, and so on, cycling through the dimensions. Each node corresponds to a rectangular region of the plane; child nodes again partition the parent region. A range query recursively visits all partitions of the tree that intersect the query range and returns all the visited nodes in the range; the worst-case query time is $O(n^{1-1/d})$ in d dimensions.

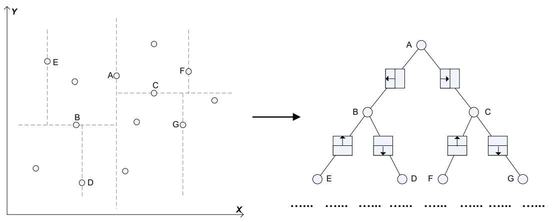


Fig. 1. kd-Tree Data Structure

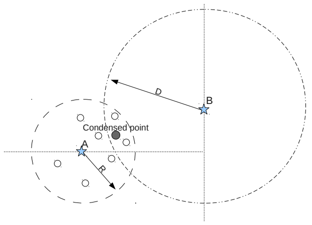


Fig. 2. Condensed kd-Node

A balanced kd-tree can be constructed by recursively inserting the median node within each region into the tree. To expedite tree construction, we use fast randomized construction to insert the nodes. Our observations show that a balanced kd-tree can be constructed by inserting accelerometer readings in sample order, since the data oscillates for typical in-network sensing tasks. As a result, the pseudo-randomized construction runs in $O(n \log n)$ time.

3.2 Nearest Neighbor Classification

We first describe the process of nearest neighbor classification in an uncondensed kd-tree. A kd-tree can be used as a classifier using k-nearest neighbor search in the pattern space, where each dimension of the tree corresponds to a dimension within the pattern space. In our case, points belonging to different classes are stored in the same tree. We begin by abandoning the standard requirement of a *fixed* k-neighbor search. Instead, we introduce a dynamic neighborhood mechanism: We define D as the *neighborhood threshold*. A node j is a neighbor of node i iff the Euclidean distance from node i to node j is less than D. For a node x, we define the magnitude $m(c_i)$ of class c_i as the number of neighboring nodes associated with c_i, weighted by their respective distances from x. The likelihood of a point belonging to a class is proportional to the magnitude of the class within its neighborhood. Let $P(c_i|x)$ denote the posteriori probability that point x belongs to class c_i, and d_p denote the Euclidean distance from point x to some neighboring point p. The number of neighboring points associated with class c_i is proportional to $P(c_i|x)$, and the distance d_p is inversely proportional to $P(c_i|x)$. Intuitively, the weight can be defined as $\frac{1}{D+d_p}$, where the inverse proportionality is linear. To exaggerate the proportionality, we impose an exponent of 2 as a penalty on d_p. As a result, we must add an exponent of 2 on D. Using D^2 as the numerator, the weight function is confined to the range [1/2, 1]. We can evaluate the magnitude of each class by taking the summation of neighboring nodes associated with the class, appropriately weighted:

$$P(c_i|x) \sim m(c_i) = \sum_p w(c_i) = \sum_p \frac{D^2}{D^2 + d_p^2} \tag{1}$$

In a kd-tree, nearest neighbor classification is an alternative form of a range query, which has worst-case $O(n^{1-1/d})$ running time. Classification begins with the *find-node* operation. *Find-node* performs an inexact search to find the query point in the tree; it stops at the node which defines the minimal region containing the query point. Call this node Q. Next, a breadth first search of the nearest neighbors of the query point is performed, beginning from Q. To support breadth first search, we use back links between children and parents. The search traverses the tree using normal breadth first search. We use neighborhood threshold distance D to eliminate branches that are out of range. The search process runs in $O(n^{1-1/d})$ time in d dimensions, the same as the range query. This running time is acceptable with a small number of nodes in a low-dimensional space. We will return to this point in a later section.

3.3 Condensed kd-Tree

Due to the memory limitations of common sensor nodes, it is typically infeasible to construct a complete kd-tree using raw accelerometer data. For example, if we collect 2000 samples, and each tree node requires 12 bytes, the tree will consume approximately 24KB – far beyond the available memory of the MSP430 (10KB). To overcome this, we introduce a condensing technique.

Table 1. kd-Node Data Structure

Domain	***Attribute***	***Description***	***Modified***
key_value[ ]	*int [D]*	invariant key values	Original
cond_value[ ]	*float [D]*	condensed mean values	Augmented
count[ ]	*int [C]*	count of condensed points of each class	Augmented
left	*struct kdNode **	left child pointer	Original
right	*struct kdNode **	right child pointer	Original
parent	*struct kdNode **	parent pointer	Original

To reduce the size of the tree, a *merge* operation is introduced. We merge each new node with an existing node if the new node is within the *condensing radius* of the existing node. The node structure is shown in Table 1: *key_value* stores the sample values originally used to define the node. *left*, *right*, and *parent* store pointers to the left child, right child, and parent, respectively. To support condensing, we augment each kd-tree node with two new fields: *count* is an array that stores the frequency of class membership for the samples merged with the node. The *cond_value* field stores the arithmetic means of all sample values represented by the node. The original sample values (*key_value*) serve as invariant keys in the condensed tree. This invariant property is necessary since *cond_value* cannot be used as a key in the search since it may change after a merge operation.

We construct the tree by inserting the data samples in sample order. The *insert* operation traverses down from the root to the leaves. The new node will be merged if it is within the condensing radius of an existing node's key values. The *merge* operation updates the condensed mean values stored by the existing node and updates the *count* element of the associated class. If a *merge* cannot be performed, the *insert* operation proceeds as usual. In either case, *insert* takes $O(\log n)$ time. As discussed later in Section 4.1, it requires little time to search in our condensed kd-tree.

Figure 2 illustrates the *merge* operation. Point A denotes a pair of invariant key values in the tree; R is the condensing radius. The hollow points around A represent the values associated with A prior to insertion. The condensed values, shown as a filled circle, are adjusted after every merge. In the condensed kd-tree, the key values maintain the tree property, while the condensed values represent the mean of all values merged with the node. Since the condensing radius limits the allowable distance of merge candidates, the condensed values will not be pulled outside the condensing radius, which avoids reconstruction of the tree.

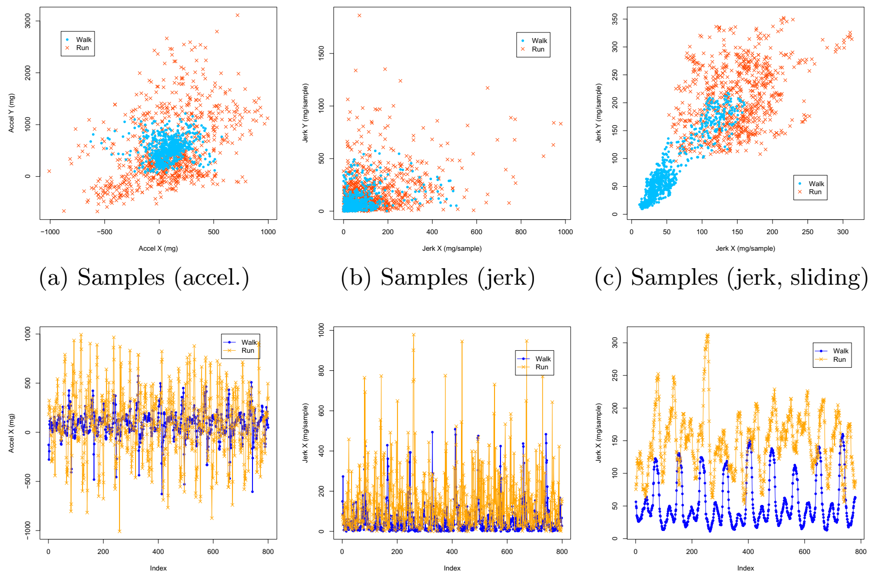
To accommodate potential "border crossings" induced by the merge operation, the classification operation must be adapted. We again use inexact search to find the node containing the most similar key values, and then begin a breadth first search to query the neighboring nodes in range. The search is based on key values. However, the associated condensed values are likely to be different, but not by a distance greater than the condensing radius R. If a condensed point has been skewed into the neighborhood threshold of a given node, while the key values are out of range, it is possible to miss a neighboring node during the search process. Figure 2 illustrates this situation. The condensed values for node A are represented by a filled circle in the graph; B is the invariant key of another node, and D is the neighborhood threshold. When a breadth first search reaches B (prior to A), A and its subtree will not be visited since it is out of range of the neighborhood threshold. However, the condensed values should be included when calculating the magnitude since it is in range. To overcome this, we use $D + R$ as the neighborhood search range instead of D during an inexact search query. The classification still runs in $O(n^{1-1/d})$ time. The magnitude function must also be revised to consider the number of merged nodes associated with class c_i, denoted by Φ_{c_i}:

$$P(c_i|x) \sim m(c_i) = \sum w(c_i) = \sum \frac{\Phi_{c_i} D^2}{D^2 + d_p^2} \tag{2}$$

To improve accuracy, R should be less than D. At the same time, if R is too small, the *condensing ratio* of the tree, i.e., one minus the ratio of the condensed size to the original size, will be low. Considering the tradeoffs, we choose the condensing radius R to be less than or equal to half of the neighborhood threshold D in our case studies. With condensing, the size of the trees in our case studies were reduced by more than 95%, while the classification accuracy was reduced by less than 10%.

3.4 Sample Preprocessing

While this approach is suitable for the target hardware platforms, raw accelerometer data is often difficult to classify directly. Consider, for example, attempting



(a) Samples (accel.) (b) Samples (jerk) (c) Samples (jerk, sliding)

(d) Acceleration (X-axis) (e) Jerk (X-axis) (f) Jerk (X-axis, sliding)

Fig. 3. The Impact of Preprocessing

to determine whether a target is walking or running based on accelerometer data collected from a sensor carried by the target. Figure 3 shows training data and corresponding preprocessed data collected from a simple trail: The user carried a sensor node in his pocket, training the classifier by walking and running for several seconds. The original acceleration readings are represented in 2 dimensions in Fig. 3a. A large number of samples are "mixed together" in the middle of the plane since the arithmetic mean of a series of vibration data settles at a fixed point, as shown in Fig. 3d. Meanwhile, the standard deviation of the data across the classes is relatively large, scattering the readings throughout the space. Since the kd-tree classifier uses neighboring samples to classify incoming data, the classifier will be error-prone if it is generated on raw accelerometer readings. For this case, the accuracy for walking events is below 50% (when tested on the training data).

To construct an accurate classifier, we must separate the data centers of the two classes. More precisely, the goal is to separate the arithmetic means and decrease the standard deviations across the two groups. For this purpose, we use *jerk*, the rate of change in acceleration: $j = \mathrm{d}\boldsymbol{a}/\mathrm{d}t = \mathrm{d}^2\boldsymbol{v}/\mathrm{d}^2t$.

By transforming the raw accelerometer readings to absolute values of jerk $|j|$, we can separate the arithmetic means of the two groups, as shown in 3e, making it possible to construct a proper classifier. As shown in Fig. 3b, fewer samples from different classes are tangled together.

However, even if the arithmetic means are separated, high standard deviations may cause the two groups to overlap, as shown in Figs. 3b and 3e. The

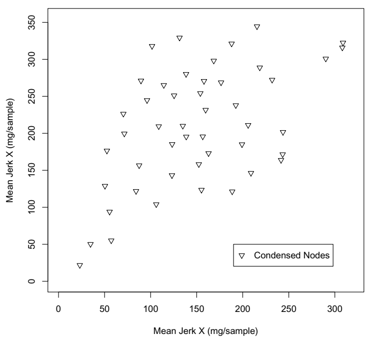


Fig. 4. kd-Tree (condensed)

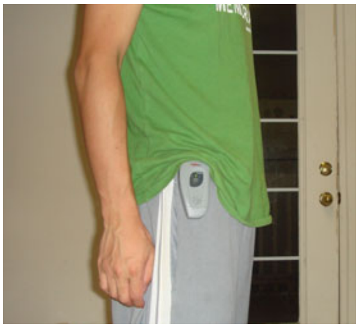


Fig. 5. Sensor Carried by User

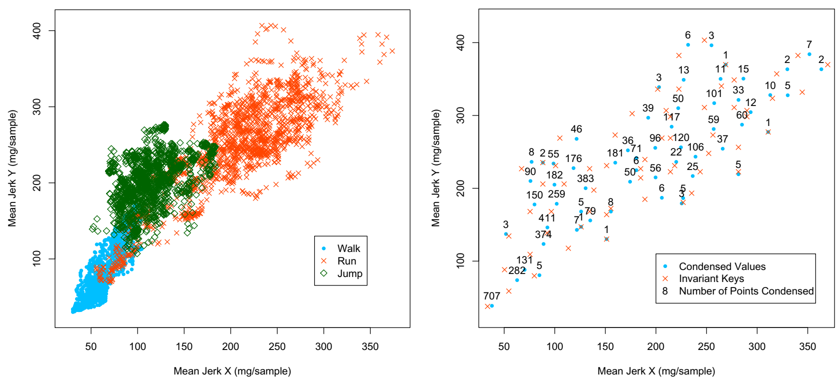
performance of the classifier can be further improved by computing the mean value of jerk over a fixed size moving window. Consider, for example, a window of size 40, corresponding to a 400 ms period when the sampling rate is 100Hz. This scenario is illustrated in Figs. 3c and 3f. The standard deviation of the jerk data is smoothed, further separating the data groups. Indeed, the two groups are almost completely separated, and therefore, the kd-tree classifier achieves accuracy above 90% in this trial. In general, the groups can be further separated by increasing the window size. However, this increases the effective sampling period. We typically choose the size in the range of 40 to 80. During preprocessing, we store jerk data in a circular queue; the arithmetic mean can be updated in $O(1)$ time. This presents a general preprocessing step for constructing classifiers on roughly periodic data with close arithmetic means and large standard deviations.

Using this preprocessed dataset, we constructed a condensed kd-tree, illustrated in Fig. 4. Each point represents a set of condensed values. We now consider the implementation and performance.

4 Case Studies

4.1 Case Study 1: Human Movement

The goal of the first application case study is to identify the *walking*, *running*, and *jumping* activities of a carrier. As introduced in Section 1, detecting human movements is a common task for wearable sensor systems [6, 8, 10]. In this case study, we use Tmote Invent sensor nodes, each equipped with a 2-axis accelerometer. The accelerometer provides measurements in the range of $\pm 5g$ in the X-Y plane of the device. As shown in Fig. 5, the carrier can put the sensor node into a pocket, or hang it on a neck strap. The orientation of the device is not required to be vertical or horizontal. The carrier simply needs to make sure that orientation is fixed across training and detection. If the orientation changes, the classifier must be retrained.

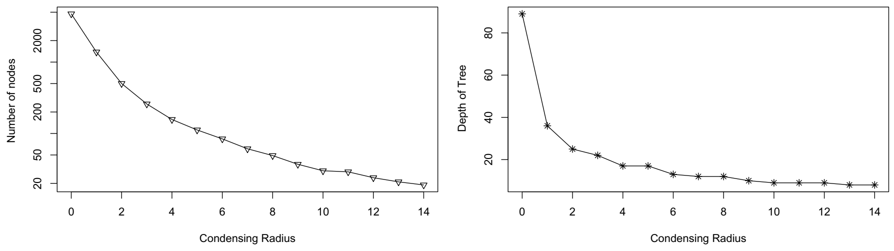


(a) Preprocessed Samples (b) Condensed Tree

Fig. 6. Condensed kd-Tree Classifier (condensing radius = 7)

Implementation of the Condensed kd-Tree Classifier. We implemented the condensed kd-tree classifier on a Tmote Invent to detect human movement. To start the training phase, the user clicks the function button on the device. The sampling rate is set to 100Hz, as recommended by the Tmote Invent manual. For each class, the training phase collects 1500 samples (15 seconds). The node computes jerk using the sliding window preprocessing technique described in Section 3.4; jerk samples are directly inserted into the tree. When sampling for a given class is complete, the device waits for a button click event to trigger classification for the next class. When all the training tasks are complete, the device transitions to the *online phase*, in which unknown data samples are classified. In our experiment, the classifier is triggered by the vibration detection module on the Tmote Invent. Once the node is triggered, it preprocesses the acceleration samples through the moving window to generate jerk data. The tree classifies the jerk data and yields the final classification result, which may be sent back to a basestation or stored on the node.

In our experiments, there are 1500 jerk samples for each class. Thus, 4500 samples are inserted into the tree. Figure 6a shows the preprocessed jerk samples for walking, running, and jumping events with a sliding window size of 50. Figure 6b shows the invariant keys and aggregate values stored in the condensed kd-tree. The cross points denote the invariant keys, and the circles denote the condensed values, which have been skewed away from the invariant keys. The number associated with each node denotes the total number of samples it represents. If we constructed this tree without condensing, it would contain 4500 nodes with 89 levels. But as shown in Fig. 6b, with a condensing radius of 7, the number of nodes has been decreased to 61, and the depth of the tree has been reduced to 12. The tree uses only 1,464 bytes of RAM – suitable for typical sensor nodes. In this case, the condensing technique saves more than 97% in memory space.



(a) Node Count (b) Tree Depth

Fig. 7. Impact of Condensing Radius on Tree Size

During the online phase, each jerk sample is used as a query point for the nearest neighbor search. The number of neighbors searched is dynamic, based on the condensing radius and neighborhood threshold. We tested condensing radius values from 1 to 14. The neighborhood threshold was set to slightly larger than twice the condensing radius. The nearest neighbor search uses the magnitude function from Section 3.3 to evaluate class membership.

Figure 7 summarizes the impact of the condensing radius on the size and depth of the generated tree. As shown in Fig. 7a, the number of nodes required to represent the classifier is in excess of 1200 with a condensing radius of 1. It drops to less than 100 when the condensing radius is set to 6, and less than 30 when the condensing radius is larger than 10. (Note that the scale on the y axis is logarithmic). Figure 7b shows that the depth of the tree also decreases significantly with increasing condensing radius. As a point of reference, tree depth is 25 with a condensing radius of 2; it decreases to 12 with a condensing radius of 7. As a result, the speed of the nearest neighbor search is dramatically increased (since the search traverses each node at most once).

We are also interested in the relationship between the size of the training dataset and the size of the resulting tree. Figure 8 summarizes the relationship. A total of 6400 training samples were inserted into the tree; the tree size was recorded after every 100 samples. The condensing radius was set to 7. The graph shows that the rate of increase in tree size decreases significantly with sample count. Indeed, the tree structure becomes relatively stable beyond 5000 samples. This feature allows the tree to be trained with large datasets. In addition, it provides the potential to accept new training data in the future to improve accuracy.

We next evaluate the relationship between condensing radius and accuracy. As shown in Fig. 9, overall classification accuracy decreases only slightly with increasing condensing radius. The average rate of decrease in accuracy is smaller than 1% for a unit increase in condensing radius. The impact does, however, vary among classes. For example, classification accuracy for running events decreases by 21% when the condensing radius is increased from 1 to 14, but accuracy for jumping events actually increases by 9% from 2 to 14. The explanation for this is that the tree loses granularity as the condensing radius is increased. As shown

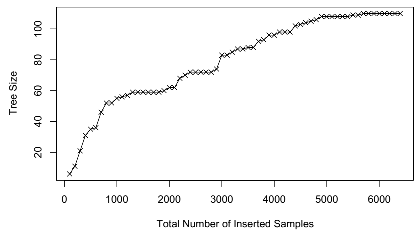


Fig. 8. Tree Size vs Sample Count

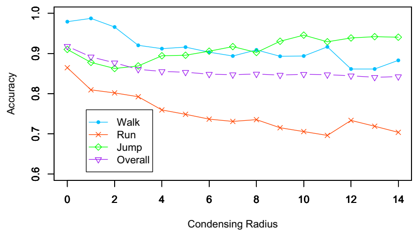


Fig. 9. Accuracy vs Cond. Radius

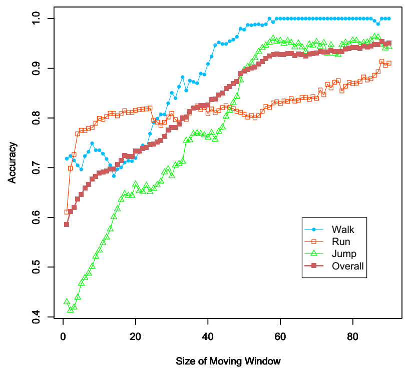


Fig. 10. Window Size vs Accuracy

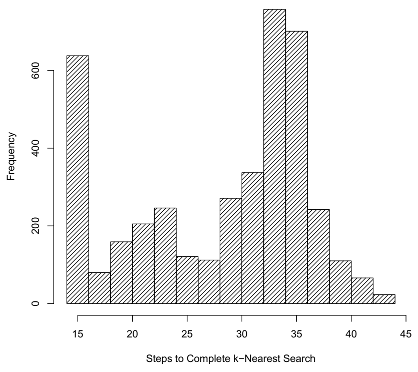


Fig. 11. Classification Speed

in Fig. 6a, the data points associated with running events are inbetween the data points associated with walking events and jumping events. At the boundary of two classes, a small set of data points are often "tangled" around the boundary. These points may be merged to a single condensed point. The condensed value is based on the average of all values in the region, and is therefore biased against the minority class in that region. The larger the condensing radius, the larger the number of nodes that may be merged with a condensed node, and by consequence, the larger the potential skew. The accuracy along the boundary may decrease as a result.

Considering the tradeoff between efficiency and accuracy, choosing an appropriate condensing radius is a key consideration in this approach. In this case study, values between 5 and 8 appear to be good choices.

We also investigate the relationship between window size and classifier accuracy. Since the standard deviation of the sample data is reduced under the condensing operation, we evaluate the impact without condensing. The results are summarized in Fig. 10. Classification accuracy is plotted for window sizes varying from 1 to 90. The sliding window preprocessing technique improves accuracy significantly. The accuracy increases with window size up to a given point. The accuracy reaches approximately 92% in this case. However, larger windows require longer sampling times and larger buffer sizes.

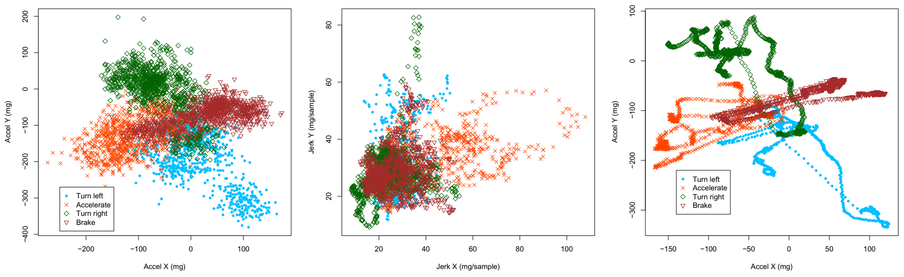
We now consider classification speed using the condensed kd-tree. We tested 4000 random samples in a condensed kd-tree constructed using the training data collected for this case study. The condensing radius was set to 7, and the neighborhood threshold was set to 15. Fig. 11 summarizes the results. The X-axis denotes the number of traversal steps taken during the nearest neighbor search; the Y-axis shows the frequency over the trial. Note that a search includes two phases – finding the target node and finding the nearest neighbors. The slowest classification takes no more than 45 steps; the fastest takes only 15. Most take 33-36 steps, or 15-16 steps. The average number of steps is 28.6. This operation is fast enough to be executed on typical sensor nodes.

The kd-tree classifier offers advantages along several key dimensions, such as accuracy, speed, storage efficiency, and retraining. Specifically, the kd-tree classifier achieves accuracy above 85%, and the classification speed of the condensed kd-tree is fast enough for typical sensing scenarios. Further, the training phase requires limited memory by avoiding the use of a large buffer for storing the training samples – as would be required, for instance, in a Bayesian classifier [20]. In the classification phase, it uses less than 2.4KB to store the tree. Most interesting, the tree can be further trained without any reconstruction. The number of nodes increases slowly with an increase in samples.

4.2 Case Study 2: Driving Events

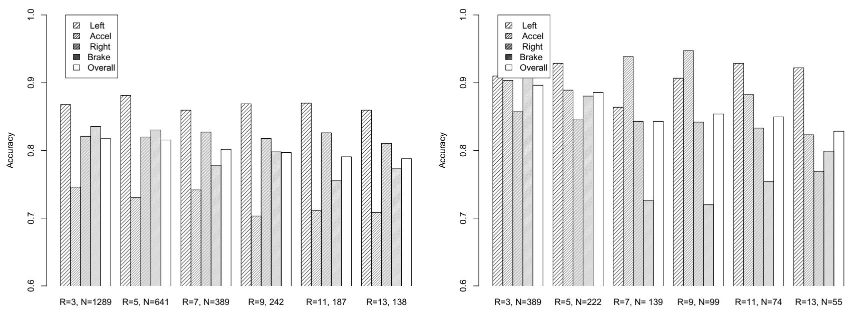
The second application case study explores the detection of driving events, again using the accelerometer on the Tmote Invent. A similar effort is discussed in [19], which centers on detecting road conditions, but without the use of a formal classifier. Our goal is to detect four basic actions: *accelerating*, *braking*, *turning left*, and *turning right*. The hardware setup is similar to the last case study. The Tmote Invent is installed inside a car; again, the orientation of the node is not relevant. The sampling period in the training phase was set to 2 seconds (200 samples); 5 training periods were performed for each event.

We applied the same preprocessing and classification techniques as in the first case study. However, we discovered that our preprocessing techniques were not suitable for the driving scenario since the absolute value of jerk is similar for most of the target classes. Figure 12b shows the preprocessed jerk samples.



(a) Acceleration samples (b) Preprocessed jerk (c) Preprocessed accel.

Fig. 12. The Impact of Preprocessing (Driving Events)



(a) No Preprocessing (b) Accel. Preprocessing

Fig. 13. Accuracy and Size of kd-Tree Classifier

Many of the data points are tangled together in the left part of the graph, especially for data points associated with turning and braking events. As a result, the classifier cannot correctly classify the preprocessed data. We analyzed the problem and found that the most significant difference among the samples from different classes is the *direction* of acceleration, which is not considered in our first approach. Hence, instead of using absolute jerk, a scalar, we used the original data, which contains direction information, to construct the classifier. Figure 12a shows the original acceleration samples. Since the acceleration directions of the four classes are different on the X-Y plane, the sample points are scattered in four parts of the graph. As a result, the generated classifier is capable of classifying the events, with high accuracy, using the original data. Figure 13a shows the accuracy of the condensed classifier using the original samples. We considered condensing radius values between 3 and 13. Without any preprocessing, the tree achieves accuracy above 78% with a condensing radius less than or equal to 11.

To further improve the accuracy of the classifier, we introduce an alternative preprocessing technique. Recall that computing the arithmetic mean across a moving window can be used to decrease the standard deviation of the data, "smoothing" the irregular vibration. Without converting the acceleration samples to jerk, we directly process the data through the moving window. As a result, the overlapping areas between different classes are significantly reduced in size. Figure 12c shows the preprocessed acceleration samples. After preprocessing, the acceleration samples appear as smooth curves. Figure 13b summarizes the accuracy using preprocessed acceleration data. Classification accuracy is improved by 5% on average, yielding overall accuracy above 85% with a condensing radius less than or equal to 11. By reducing the standard deviation, the samples are less scattered in the pattern space. Consequently, the tree size is reduced. As shown in Figs. 13a and 13b, for each condensing radius, the tree size is significantly larger without preprocessing. As a point of reference, the tree size is reduced to 74 with a condensing radius of 11, and the overall accuracy is still above 85%.

Based on the case studies considered, using jerk samples to construct a classifier is more effective for scenarios where the difference among events is based largely on undirected vibration. In these scenarios, the arithmetic means of the classes formed based on the original samples are close, and the standard deviations are typically large. Using preprocessing to transform these samples to jerk, the arithmetic means can be separated, and the standard deviations decreased. In contrast, it is preferable to construct a classifier using acceleration data if the main difference among events is the direction of acceleration. The sliding window preprocessing technique can be applied on both scenarios to improve the performance, but without computing jerk samples in the second scenario.

5 Conclusion

While machine learning techniques offer a number of advantages in the context of sensor-based event detection, their application presents a challenge for in situ scenarios. Existing techniques are resource-intensive, precluding direct implementation on mote-class platforms. In this paper, we described a new technique that supports both training and classification on resource-lean devices.

Condensed kd-tree classification is a novel classification method that uses an enhancement of a standard kd-tree as the underlying representation structure. A dynamically-scoped nearest neighbor search technique is used to classify incoming data samples based on the distance-weighted categorizations of corresponding neighbors. The representation allows developers to adjust the tree size to accommodate memory-limited hardware without a significant loss in classifier accuracy. We considered pre-processing enhancements to the classifier and evaluated its performance in two representative case studies.

The classification method supports fast classification, achieves high accuracy, requires little memory, and supports in-network retraining. It is suitable for a wide range of sensor network applications.

Acknowledgments. This work is supported by the National Science Foundation award CNS-0745846.

References

1. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Communications Magazine 40(8), 102–114 (2002)
2. Breiman, L., et al.: Classification and regression trees (1984)
3. Burchfield, T.R., Venkatesan, S.: Accelerometer-based human abnormal movement detection in wireless sensor nets. In: HealthNet 2007, pp. 67–69. ACM, New York (2007)
4. Cover, T., Hart, P.: Nearest neighbor pattern classiflcation. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
5. Forney, G.: The viterbi algorithm. In: Proceedings of the IEEE, vol. 61, pp. 268–278. IEEE, Los Alamitos (1973)
6. Ganti, R.K., Jayachandran, P., Abdelzaher, T.F., Stankovic, J.A.: Satire: a software architecture for smart attire. In: MobiSys 2006, pp. 110–123. ACM, New York (2006)

7. Grother, P.J., Candela, G.T., Blue, J.L.: Fast implementations of nearest neighbor classifiers. Pattern Recognition 30(3), 459–465 (1997), `http://www.sciencedirect.com/science/article/B6V14-3SNTGXH-T/2/208700cc812e769ea72e11e348dd2dbd`
8. Györbíró, N., Fábián, A., Hományi, G.: An activity recognition system for mobile phones. Mobile Network and Applications 14(1), 82–91 (2009)
9. He, J., et al.: Real-time daily activity classification with wireless sensor networks using hidden markov model. In: EMBC 2007, pp. 3192–3195. IEEE, Los Alamitos (2007)
10. Henk, C.R., Muller, H.: Context awareness by analysing accelerometer data. In: The 4th Int. Symp. on Wearable Computers, pp. 175–176. IEEE, Los Alamitos (2000)
11. Kim, S., et al.: Wireless sensor networks for structural health monitoring. In: SenSys 2006, pp. 427–428. ACM, New York (2006)
12. Kim, S., et al.: Health monitoring of civil infrastructures using wireless sensor networks. In: IPSN 2007, pp. 254–263. ACM, New York (2007)
13. Kim, Y., et al.: An efficient scheme of target classification and information fusion in wireless sensor networks. Personal Ubiquitous Comput. 13(7), 499–508 (2009)
14. Levis, P., et al.: Tinyos: An operating system for sensor networks. In: Ambient Intelligence, pp. 115–148. Springer, Heidelberg (2005), `http://dx.doi.org/10.1007/3-540-27139-2_7`
15. Logan, B.: Mel frequency cepstral coefficients for music modeling. In: International Symposium on Music Information Retrieval (2000)
16. Lorincz, K., et al.: Mercury: a wearable sensor network platform for high-fidelity motion analysis. In: SenSys 2009, pp. 183–196. ACM, New York (2009)
17. Lu, H., et al.: Soundsense: scalable sound sensing for people-centric applications on mobile phones. In: MobiSys 2009, pp. 165–178. ACM, New York (2009)
18. Miluzzo, E., et al.: Sensing meets mobile social networks: the design, imp. and eval. of the CenceMe application. In: SenSys 2008, pp. 337–350. ACM, New York (2008)
19. Mohan, P., et al.: Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In: SenSys 2008, pp. 323–336. ACM Press, New York (2008)
20. Principe, J.C., Euliano, N.R., Lefebvre, W.C.: Neural and Adaptive Systems: Fundamentals through Simulations. Wiley, Nueva York (2000), `http://148.201.94.3:8991/F?func=direct&current_base=ITE01&doc_number=000152770`
21. Reynolds, D.A., Rose, R.C.: Robust text-independent speaker identification using gaussian mixture speaker models. IEEE Transactions on Speech and Audio Processing 3(1), 72–83 (1995)
22. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. In: SIGMOD 1995, pp. 71–79. ACM, New York (1995)
23. Wang, Y., et al.: Predicting link quality using supervised learning in wireless sensor networks. SIGMOBILE Mob. Comput. Commun. Rev. 11(3), 71–83 (2007)
24. Werner-Allen, G., et al.: Lance: optimizing high-resolution signal collection in wireless sensor networks. In: SenSys 2008, pp. 169–182. ACM, New York (2008)
25. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, San Francisco (2005), `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0120884070`
26. Xu, N., et al.: A wireless sensor network for structural monitoring. In: SenSys 2004, pp. 13–24. ACM, New York (2004)

A Mobility Management Framework for Optimizing the Trajectory of a Mobile Base-Station

Madhu Mudigonda[1], Trisul Kanipakam[1], Adam Dutko[1], Manohar Bathula[1], Nigamanth Sridhar[1], Srinivasan Seetharaman[2], and Jason O. Hallstrom[3]

[1] Electrical & Computer Engineering, Cleveland State University, OH, USA
{mudigonda.mm,trishchakri,dutko.adam}@gmail.com,
{m.bathula,n.sridhar1}@csuohio.edu
[2] Deutsche Telekom Labs, Germany
srini.seetharaman@telekom.com
[3] School of Computing, Clemson University, SC, USA
jasonoh@cs.clemson.edu

Abstract. We describe a software framework for prescribing the trajectory path of a mobile sink in a wireless sensor network under an extensible set of optimization criteria. The framework relies on an integrated mobility manager that continuously advises the sink using application-specific network statistics. We focus on a reference implementation for TinyOS. Through extensive physical experimentation, we show that the mobility manager significantly improves network performance under a range of optimization scenarios.

1 Introduction

Wireless sensor networks afford the promise of ultra-dense instrumentation of the natural and built environment for purposes of observation and control. For these networks to become integrated as part of a permanent planetary monitoring fabric, network longevity obstacles must be overcome. Several methods have been proposed to extend the lifetime of multi-hop networks, including the use of multiple sinks, improved sensor distribution, energy-balanced clustering, data mules, and other strategies. In this paper, we focus on the use of *mobile base-stations*, which periodically alter the routing structure to avoid the possibility of static bottlenecks. While introducing a mobile sink is not always possible, a number of existing applications inherently rely on mobile sinks.

Sink mobility introduces a number of questions: When should a sink move? Where should it move? How is its location shared with the routing network? How are multi-hop routes maintained in the presence of mobility? Our objective is to design a generalized framework that provides answers to these questions. To our knowledge, we are the first to consider an intelligent mobile sink that requires no a priori scheduling. Instead, a distributed mobility manager collects salient network statistics and uses these statistics to drive mobility decisions. While others have considered the problem of sink mobility in sensor networks, all but a few [7, 9] have relied exclusively on simulations. In contrast, we have developed an integrated hardware/software testbed to conduct experiments. The testbed includes a programmable mobile element and a network of 30 TelosB nodes.

We report three contributions: *(Sects. 2 and 4)* A generalized framework and reference architecture to support mobility decisions in sensor networks. *(Sect. 3)* A collection of decision metrics to support sink mobility, including (*i*) residual node energy, (*ii*) regional network congestion, and (*iii*) average relay distance. We also describe how new metrics are readily integrated into the framework. *(Sect. 5)* A prototype implementation of the framework for TinyOS.

Problem Definition. We consider a sensor network consisting of an arbitrary number of nodes, deployed arbitrarily. Each node participates in a spanning tree routing layer rooted at a base-station (sink) and communicates sampled data over this layer. Hence, the load on each device comprises (*i*) sampling sensors, (*ii*) transmitting sampled data toward the sink, (*iii*) relaying received data toward the sink, and (*iv*) updating the routing path when needed (e.g., , due to node death). For nodes closest to the base-station, (*iii*) quickly becomes the dominating factor; a small set of nodes must relay *all* sampled network data, creating a lifetime bottleneck.

While there is substantial prior work focused on introducing sink mobility to mitigate this bottleneck (detailed in Sect. 7), questions concerning the trajectory of the sink have not been completely addressed. Examples of simplifying assumptions adopted in prior work include time invariance of battery and radio performance, uniform radio propagation range, and independence of MAC delays on energy consumption [7, 11, 21]. In contrast, our work makes only two basic assumptions: The target network must provide time synchronization, and the geographic extent of the network must be known to the mobile sink.

While the most important mobility goal is ensuring uniform energy consumption across the network, other optimization metrics are also possible. A base-station might, for instance, attempt to locate itself to reduce network congestion or to overhear an interesting data stream. Hence, we define the problem as follows: **the design of a generalized framework for prescribing a sink mobility path under an extensible set of optimization criteria, in a manner responsive to real-time network conditions.**

2 System Architecture

The architecture is illustrated in Fig. 1; the corresponding component interfaces, expressed in nesC, are shown in Fig. 2.

Metric Generator. An implementation of the MetricGenerator interface runs on each node, monitoring its runtime behavior and recording a set of statistics material to the relevant optimization goal. For example, ResidualEnergyMonitor, discussed later, is a specialization of MetricGenerator that computes a real-time

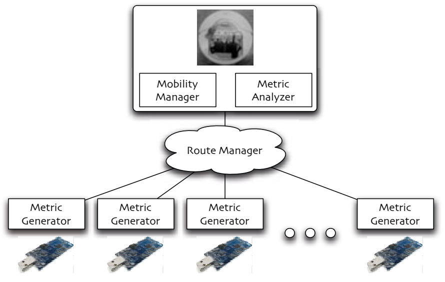


Fig. 1. Framework Architecture

```
interface MetricGenerator {
  command int getCurrentValue();
}
interface MetricAnalyzer {
  command int compare(int m1, int m2);
}
interface RouteManager {
  command void formRoute();
  command void cancelRoute();
  command int getParent();
  command int getDistance();
}
interface MobilityManager {
  command bool isMoveNecessary();
  command int getNewRegion();
  command void moveToRegion(int target);
}
```

Fig. 2. Framework Interfaces

estimate of available device energy. This value (obtained through a call to getCurrentValue()) is then transmitted over the routing layer to the sink to support its mobility decision. In many cases, the metric data can be piggybacked on standard application data packets. In Sect. 3, we describe three different metrics for which we have built generators. The descriptions there roughly correspond to implementations of getCurrentValue().

Metric Analyzer. The sink must be able to impose an ordering on this data for purposes of comparison. In the case of residual energy, the sink should move to the region where the average residual energy is *highest*, whereas in the case of network congestion, the sink should move to the region where congestion is *lowest*. To support relative valuation, an implementation of the MetricAnalyzer interface is used on the mobile device. The interface provides a compare() operation that returns 1 if m1 is *better* than m2, −1 if m2 is better than m1, and 0 if the values are indistinguishable — where the definition of *better* depends on the optimization goal. The mobility manager relies on MetricAnalyzer to determine when the local-area average of collected metric values dictates a move, and to perform pairwise area comparisons when determining where to move.

Route Manager. When the sink moves from one position to another, the routing topology must be updated to ensure node-to-sink connectivity. This process is supported by an implementation of the RouteManager interface, used both at the sensing end-points and the sink. When a move begins, the sink invokes cancelRoute() to initiate route cancelation. When the move is complete, formRoute() is invoked to reestablish the routing tree. Each end-point records its position using two state variables, the node's parent and distance from the root. RouteManager is intentionally general. Several strategies have been proposed to dynamically modify the structure of a routing tree [2, 4, 11, 12, 18]. Any of these may be used to implement the RouteManager interface.

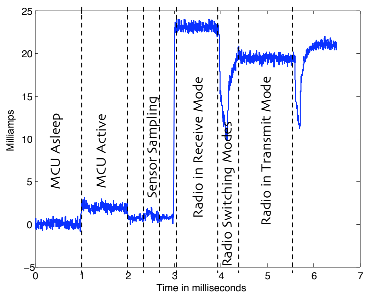


Fig. 3. TelosB Current Draw

Term	**Notation**	**Current**
Radio in transmit mode	I_{Tx}	17.4675 mA
Radio in receive mode	I_{Rx}	21.0675 mA
MCU active	$I_{McuActive}$	1.9325 mA
MCU sleep	$I_{McuSleep}$	0.0437 mA
Sensor use	I_{Sensor}	0.0061 mA

Fig. 4. Tmote Sky Current Draw

Mobility Manager. An implementation of MobilityManager is used by the sink to signal the need for movement and identify the new target location. The sink periodically polls for a movement signal using isMoveNecessary(). If a true response is received, getNewRegion() is used to compute the new sink location. This decision is based on a pairwise comparison of the metric values associated with all movement alternatives (using MetricGenerator and MetricAnalyzer). Finally, moveToRegion() directs the mobile platform to its new location.

3 Decision Metrics

Residual Energy. The first decision point we consider relates directly to network longevity — the residual energy available to each device. We adopt the *Credit-Point* (CREP) system [23] for estimating residual energy: $E_{max} = V_b \times I_b \times 3600\ Joules$. Here V_b and I_b correspond to battery voltage and capacity, respectively. Borrowing from [23], for a pair of AA batteries, the capacity is 2.2 A-Hr, with an effective voltage of 3V; we can compute the maximum energy as $E_{max} = 3 \times 2.2 \times 3600 = 23760$ J. Based on this initial energy budget, the CREP system deducts "energy points" from E_{max} for each action on the device.

The CREP model assumes a constant battery voltage, which is invalid for most, if not all, battery chemistries. To lift this assumption, we periodically sample the battery voltage using VoltageC, provided by TinyOS. The results are used to compute the energy consumed during a given period based on (1), substituting the actual battery voltage for V_b. To deduct "energy points", the monitor records the actions performed by a node during each activity period, along with the duration of each activity, focusing only on the most energy-intensive activities. For example, the monitor records the time the microcontroller was active, the time the radio spent in transmit and receive modes, the particular sensors activated, etc. Energy points are assigned empirically by measuring the current draw during each activity/configuration (a priori). A sample current plot is shown in Fig. 3; mean values are used as energy points, as summarized in Table 4. Using these values, energy points are deducted for each activity performed. No additional hardware is required to support this metric.

Network Congestion. The second metric is designed to minimize network congestion, and thus avoid message loss and energy depletion through retransmissions. To compute the level of network congestion in a given region, we aggregate multiple measures. Specifically, our congestion monitor measures the average packet reception rate (PRR), the average received signal strength (RSSI), and the average link quality indicator (LQI) along all incoming links at each node. In many cases, PRR can be measured directly. In particular, if messages are transmitted at a specific period in a given application, receiving nodes can measure the packet reception rate directly. Alternatively, in [16], the authors study the correlation between RSSI and PRR. They conclude that when RSSI values are higher than the radio sensitivity threshold (about −90dBm for the CC2420 radio), they are strongly correlated with the packet reception rate.

It is important to note that radio-level metrics, like RSSI and LQI, provide information beyond what can be gleaned from the packet reception rate alone. RSSI, for instance, is useful in measuring the noise floor within the vicinity of a node. A high noise floor increases the likelihood of congestion(-like scenarios). Bluetooth devices, when operated near a CC2420 radio, can cause the noise floor to rise as high as −25dBm. Providing access to this information enables the mobile sink to avoid such regions during intermittent periods of interference.

Average Distance to Sink. The final metric is designed to reduce the average number of transmission hops. If, for instance, the base-station is located far from the principal data source (which may vary over time), the sink should relocate closer to the source to reduce the workload on intermediate nodes. The implementation approach is straightforward. It assumes that each data packet is tagged with the distance-to-root value maintained by the publishing device. This information is then used by each node to compute a moving average over the distances of the nodes contained within its respective routing subtree.

4 Managing Mobility

Discovery Phase. We assume that the mobile base-station has no a priori knowledge of the network deployment. Instead, the sink uses the assumed network extent information to divide the deployment area into a regular grid of a desired granularity. It then proceeds to tour the region, using a beacon-based discovery process to define the membership of each grid cell. If a node is “contained” in multiple cells, it is assigned to the cell that offers the best link quality. This discovery phase may be repeated if network characteristics are fundamentally changed — if, for instance, nodes are inserted or removed.

Mobility Management Phase. Throughout the life of the network, the mobility manager is responsible for signaling the need for sink movement and determining the new target location. But the travel trajectory is not without constraints.

How far can the sink go? The maximum travel distance in a single move is limited by the duty cycle of the application. More specifically, the best performance will be achieved if the time required to complete a move is less than

```
procedure $MobilityManager$
  While collecting metrics:
    if ($alarmExpires$) then call $MetricAnalyzer$
end $MobilityManager$
procedure $MetricAnalyzer$
  Calculate average metric for $R_{current}$
  if compare($avg\_metric(R_{current}), threshold$) $< 0$ then call $Move$
end $MetricAnalyzer$
procedure $Move$
  $R_{new} := (r \in \mathcal{R} : (\forall r' \in \mathcal{R}_{reach}$ : compare$(avg\_metric(r), avg\_metric(r')) \geq 0)$
  Calculate route from $R_{current}$ to $R_{new}$
  Broadcast $cancel\_route$ message
  Perform move
  Advertise current location to initiate route reconstruction
end $Move$
```

Fig. 5. Mobility Management Algorithm

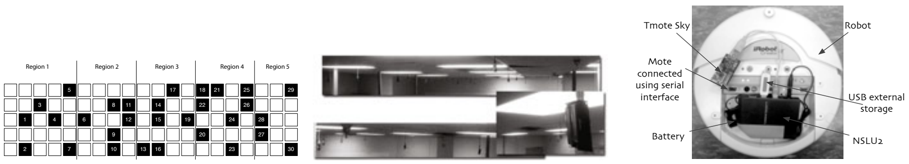
the application reporting period. Otherwise, the movement of the base-station will interfere with data collection (since the routing tree will be in transition). More important, stationary nodes are required to remain active during the transition to ensure route reconstruction. Hence, long moves can degrade system performance and limit the lifetime of the network.

To limit the reach of the sink, the mobility manager estimates the travel time associated with each path candidate based on the speed of the mobile device. The manager culls candidate paths that require travel time significantly beyond that of the application reporting period. This includes subtracting, from the allowable time, the time spent ranking the target candidates based on collected metric data, as well as the estimated time to reconstruct the routing tree.

Where should the sink go? We now consider the details of the mobility algorithm summarized in Fig. 5. While the sink is stationary, the mobility manager receives a continuous stream of metric data from across the network. At a fixed period, an *analysis alarm* is signaled to activate the metric analyzer. To synchronize sink movement with the application sleep cycle, the alarm period is a multiple of the application sleep period. The analyzer in turn compares the average of the decision metrics from its current region ($R_{current}$) to its movement threshold and determines whether to move.

If a move is required, the sink identifies the best target location (R_{new}) by comparing the metric averages of all regions within its single-step reach (R_{reach}). This comparison is realized using an implementation of compare() appropriate to the desired optimization goal. It next computes the route to the new location and broadcasts a *cancel_route* message to alert the network that it has become unrooted. The stationary nodes wait in an active state during the move and wait for the base-station's location advertisement to begin route reconstruction.

The metric analyzer may require history data to achieve optimal performance. If the analyzer simply selects the best target location within its reach during each step, it could become trapped within a set of local maxima/minima. A more sophisticated analyzer can overcome this problem. Given that the analyzer has



(a) Testbed Layout (b) Testbed Realization (c) Mobility Platform

Fig. 6. Testbed Infrastructure

access to network-wide metric data, it can compute the optimal target location and factor this information into the movement decision in each step. Hence, the analyzer can select, in each step, the region within its reach that brings the sink closer to the network-wide optimum.

5 Evaluation and Results

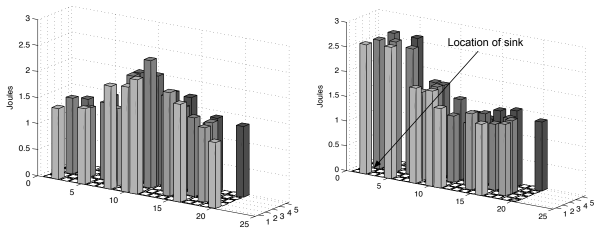
We developed a prototype implementation for TinyOS 2.x and conducted extensive experimental studies using a physical testbed. Given that our primary optimization goal was network lifetime, we compared the lifetime benefits achieved using our sink manager to other mobility management strategies.

Testbed Infrastructure. Our studies were conducted using a testbed of 30 ceiling-mounted *Tmote Sky* nodes placed in a pseudo-random fashion throughout a 1500 sq. ft. (60' x 25') area, as shown in Figs. 6(a) and 6(b). To ensure multi-hop connectivity, radio power was reduced, and each node was required to select a parent outside its grid cell. The motes were powered using standard consumer batteries; each experiment used a fresh set. The base-station uses a Tmote Sky linked to a *Linksys NSLU2* device. The NSLU2 is in turn connected to an *iRobot Create*^{TM}, shown in Fig. 6(c). As shown in Fig. 6(a), the testbed is partitioned into 5 regions. The sink's reach is limited to one region transition per step.

5.1 Energy Consumption and Lifetime

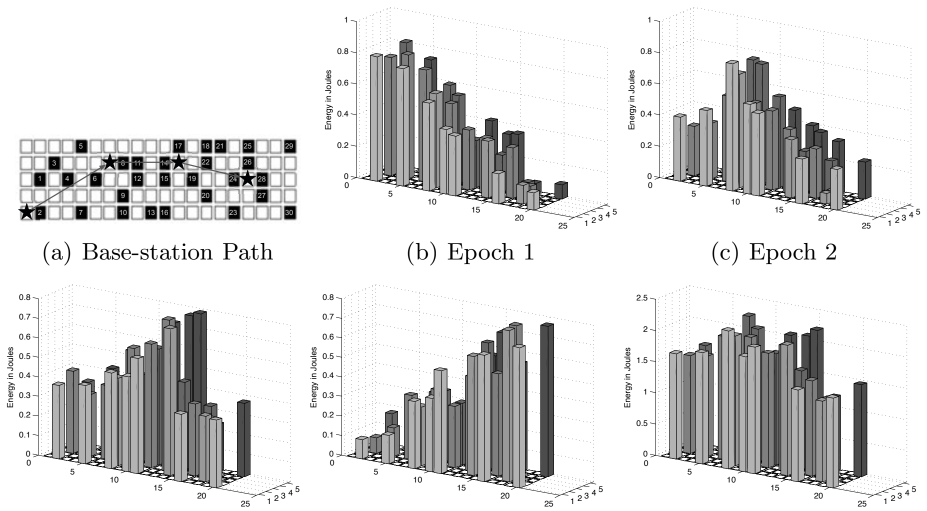
We characterize the network lifetime increase enabled by a sink under four mobility strategies: (*a*) no movement, (*b*) a fixed arbitrary path, (*c*) a fixed path computed using a linear programming formulation, and (*d*) a dynamic path guided by our mobility manager. The experiments were conducted using an application with a sleep period of 10 minutes. Each time a node wakes, it samples its local sensors and transmits the sampled data to its parent. It then waits to receive and relay each data point from its subtree before returning to sleep.

Static Base-Station. We ran two experiments with a static sink. The two were identical, except for the sink location. In the first run, the base-station was placed near the center of the network. In the second, it was placed near one end



(a) Sink (near the middle) (b) Sink (near one end)

Fig. 7. Energy Usage (static base-station)



(a) Base-station Path (b) Epoch 1 (c) Epoch 2

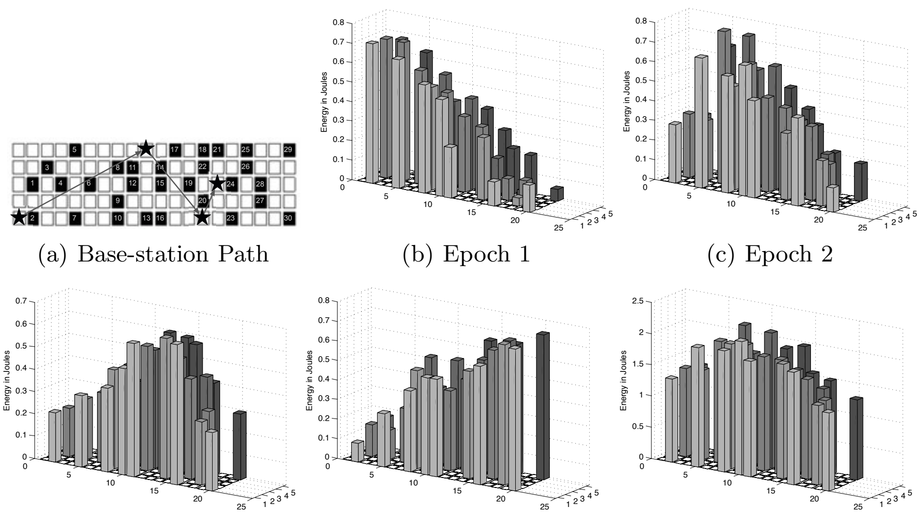
(d) Epoch 3 (e) Epoch 4 (f) Epochs 1-4 (total)

Fig. 8. Energy Usage (mobile, arbitrary path)

of the network. Each run was executed for a duration of ten hours. We measured the residual energy of each node before and after each run. Figs. 7(a) and 7(b), summarize the energy consumed by each node in the network, in Joules.

As shown in Fig. 7(a), nodes near the center of the network consumed considerably more energy than nodes near the edges of the network. The situation in the second experiment is similar, except that the peak is shifted to the network's edge, as shown in Fig. 7(b). The results are not surprising: The activity period of nodes closer to the base-station is longer since these nodes must forward all of the messages from their respective subtrees. Since the base-station is static, nodes near the base-station consume more energy over the run.

Mobile Base-Station on Arbitrary Path. We next consider a mobile base-station that traverses a pre-determined path spanning each of the regions within



(a) Base-station Path (b) Epoch 1 (c) Epoch 2

(d) Epoch 3 (e) Epoch 4 (f) Epochs 1-4 (total)

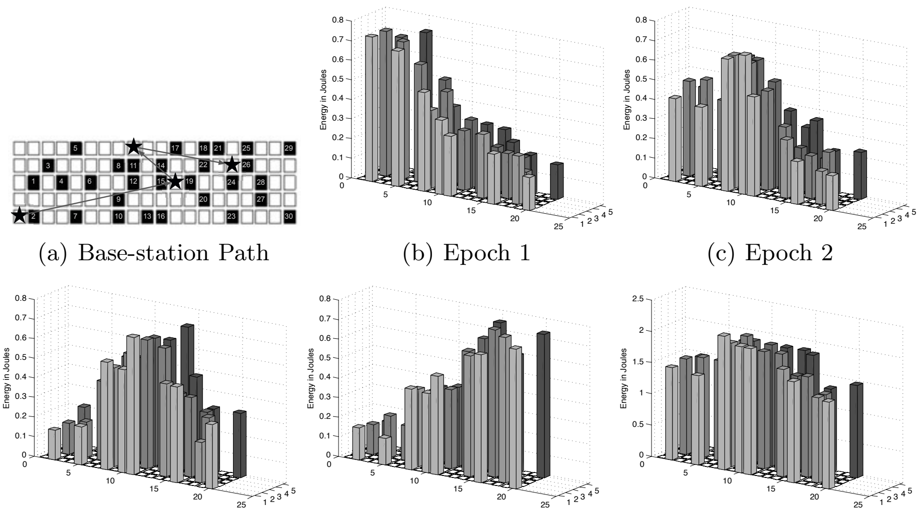
Fig. 9. Energy Usage (mobile, linear programming)

the deployment area. The device remains stationary at four points for an epoch of 2.5 hours each. The results are summarized in Fig. 8.

As the sink moves from region to region, the energy consumption trend is clearly visible in Figs. 8(b)–8(e). The combined total consumption is shown in Fig. 8(f). Although the consumption pattern is not completely uniform, it is improved over the static case. Comparing Figs. 8(f) and 7(a), the average difference between the maximum and minimum consumption across nodes is approximately $1.5J$ in the static sink case, and approximately $1J$ in the mobile case. While the result is far from optimal, it demonstrates the lifetime extension opportunities afforded by a mobile sink. The limiting factor is that the movement path ignores the residual energy available across the network. It offers no way to adapt to consumption conditions.

Mobile Sink on Pre-Computed Path using LP. Prior work in route management relies on linear programming (LP) to derive an optimal path for the mobile sink [3]. An important characteristic of the LP approach is an underlying assumption that radio behavior is time-invariant. This enables pre-calculation of the complete base-station route and corresponding sojourn times based on the transmission range of each sensor and the sink, respectively. We applied a standard LP formulation to our setup to obtain a pre-calculated sink path and compared the performance of the LP approach to our algorithm. Here we describe the important elements of the LP formulation, beginning with the objective function: *maximize* $\{\ min_{Node_i}\ \{residual\ energy\ at\ node\ i\}\ \}$.

The residual energy of each mote can be computed as the difference between the initial energy and the energy consumed during each sojourn period of the



(a) Base-station Path (b) Epoch 1 (c) Epoch 2

(d) Epoch 3 (e) Epoch 4 (f) Epochs 1-4 (total)

Fig. 10. Energy Usage (mobile, mobility manager)

mobile sink. Thus, given the energy consumed at each node while the sink is at a location k, it is easy to compute the optimal sink path.

However, computing the energy consumed at each sensor node is non-trivial. The residual energy available at a given device depends on its data relay activity. Simultaneously solving for the optimal inter-node routes and the sink traversal path is generally not possible in a linear program. Hence, we partition the program into two components and run them serially. The behavior of each node is assumed to be time-invariant; we pre-compute the optimal multi-hop routes from each node to every potential sink location. For each sink site k, determining the multi-hop route that yields the least energy consumption involves solving a supplementary LP with an objective of *minimize* { *maximum* {*energy consumed at each node i when sink is at site k*} }. On solving the supplementary LP and passing the output to the primary residual energy maximization LP, we obtain the desired route, which is optimal when radio behavior is time-invariant. Note that the ordering of the target locations is only influenced by the reach constraint imposed on the sink in each step. Thus, using the observed energy depletion rate and the reach information in our testbed, we compute the LP-based sink route.

Figure 9 shows the results across four epochs, each of 2.5 hours in duration, when the sink uses this computed path. Figures 9(b)–9(e) show the energy consumed in each epoch; Fig. 9(f) shows the total energy consumed.

Mobile Base-Station on Dynamic Path. Finally, we study the performance of the mobility management framework. In this set of experiments, the base-station is deployed at an arbitrary initial location without a pre-defined traversal path. The mobility management framework is used to collect residual energy

Table 1. Standard Deviation of Power Consumption

Sink Location	**Mean (J)**	**Std Deviation (J)**
Static (Middle)	1.62	0.4393
Static (End)	1.75	0.5981
Mobile (Arbitrary)	1.56	0.3270
Mobile (LP-computed)	1.49	0.2330
Mobile (Dynamic)	*1.43*	*0.1329*

data and to inform the sink's trajectory to maximize residual energy across the network. Every hour, the sink determines whether the regional residual energy has fallen below the required threshold. If so, the mobile sink initiates a move. It first computes the average amount of residual energy available in each region within its reach. The sink then determines the new target location and directs the robot to move to the corresponding coordinate location.

Again, the experiment was executed for ten hours. Note that it does not matter where the sink is placed since the mobility manager uses network measurements to guide its path. The experiment resulted in a total of three moves, dividing the run into four epochs. The results are summarized in Fig. 10. It is important to emphasize the low degree of variability across the network.

Comparing Mobility Strategies. To quantify the uniformity of energy consumption across mobility strategies, we compare standard deviations in Table 1. When the static sink is placed at the center of the network, the standard deviation in residual energy is low. Perhaps surprisingly, the opposite is true when the sink is placed at one end of the testbed. The explanation is straightforward: The nodes in the testbed form their routing paths based on network connectivity. When the static sink is placed at one end of the network, the nodes that are at the other end are at a distance of five hops from the sink. However, when the sink is at the center, the nodes that are furthest away are only three hops; the resulting difference in load is smaller.

The standard deviation in energy consumption is smallest in the case of the dynamically computed path. This shows that our mobility decision engine is effective in achieving the intended goal — ensuring that all nodes in a sensor network deplete their energy reserves at approximately equal rates. Further, the mean energy consumption is also reduced when the sink is moved along the path computed by our mobility manager.

Effective Lifetime. The goal of engineering an intelligent sink is to extend effective network lifetime. Figure 11 shows a comparison of expected node lifetimes using various mobility management strategies. These are estimates of how long each device will stay active based on battery capacity and average consumption rates; network dynamics will obviously play a role in determining the actual lifetimes. The arbitrary path strategy does a poor job of ensuring uniformity. As expected, the LP-computed path reduces variability, as does our dynamic

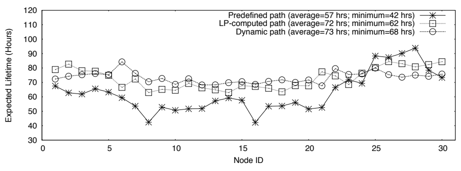


Fig. 11. Lifetime Extensions across Strategies

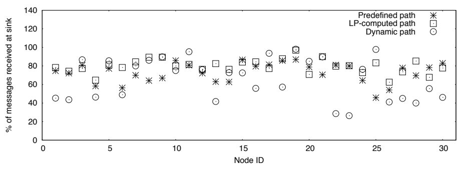


Fig. 12. Network Throughput

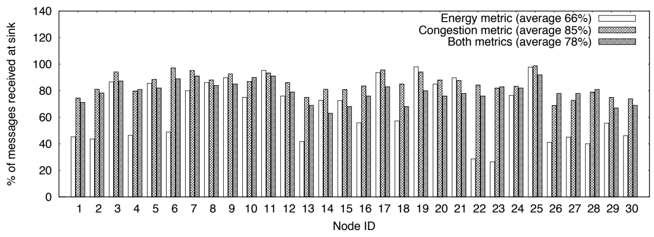


Fig. 13. Network Throughput (lifetime, congestion)

strategy. Notice, however, the expected lifetime of the first node to die in each of these cases; there is a significant difference. With a pre-defined path, the first node dies in 42 hours, whereas with the LP-computed path, the first node dies at 63 hours. Using our mobility manager, the first node dies after 68 hours.

Mobility Overhead. There is a cost associated with each transition. When the sink moves from one location to another, routes from the stationary nodes to the sink must be recomputed. In our case, we use a naïve strategy, canceling all routes and reforming the tree. Over the experiments we ran, we observed that the energy cost for the stationary nodes is about 0.1 J for each sink transition. Further, our protocol introduces an extra cost for sending residual energy data (4 bytes per message). The overhead is justified by the significant benefit that changes in the network topology can offer.

Throughput. Finally, we consider the throughput of the network as a validation measure to ensure that the application is functioning properly. While our initial experiments exhibited poor yield, we were able to improve the yield using a supplementary radio layer focused on reducing link quality variation among nodes [6]. Figure 12 shows that most nodes deliver 60% to 80% of their messages.

5.2 Network Congestion

We also studied the impact of the network congestion decision metric. The experimental setup (length, application) was the same, except that the mobility manager used NetworkCongestionMonitor as the metric generator. In Fig. 13, we compare the average throughput from each node when using the network congestion metric as compared to the residual energy metric. When the sink makes mobility decisions based on network congestion, the average throughput increases from 66% to 85%. Note that we did not consider network longevity, focusing exclusively on improving network throughput.

We next repeated the experiments, applying both metrics. The primary metric was network congestion; residual energy was used as a secondary metric in the event of ties. In this case, the average throughput was approximately 79%, and the lifetime of the first node to die was 59 hours. In the future, we expect to further investigate simultaneous optimization strategies using multiple metrics.

6 Discussion

Extending to Other Metrics. The prototype implementation of the mobility management framework demonstrates the utility of our approach in managing a mobile sink. Using this implementation, we have conducted significant evaluation studies to verify that the approach achieves uniform energy consumption across the network. The primary extension point for using the implementation in settings other than a testbed environment is the use of other decision metric components. Although we have not presented complete evaluation results here (primarily for lack of space), we have experimented with this possibility using the network congestion monitor and the message distance monitor. The benefit afforded by these metrics is easy to see: If the sink is located in a congested area, the throughput of the network is bound to suffer; and if a majority of messages must travel a long distance, throughput is again going to be affected.

Our mobility management framework can be viewed as a harness for developing specific strategies for controlling a mobile sink. For example, in [9], the authors use a mobile sink to collect data from a sensor network with the aim of reducing the number of multi-hop data transmissions. The sink can travel at varying speeds depending upon how much data it can buffer. This system can be implemented using our framework by designing a metric analyzer component sensitive to buffer size, which determines both a target speed and location for the sink. One of the directions for our future development is to develop a generic testbed that can be available for testing such mobility protocols.

Sophistication in Metric Analysis. A potential problem with performing simple pair-wise analysis in the metric analyzer is network partitioning. Consider a scenario in which the sink is in a given region of the network, and a small number of nodes bridge its current reach with the rest of the network. If these nodes die, the sink may deplete the regions in its current reach and declare the lifetime of the sensor network to be over. With a more careful analysis, the sink can detect that the throughput has dropped in an unpredictable manner. This recognition can trigger another discovery traversal to rebuild the sink's view of the available nodes.

As another example, we observe that there is a correlation between the in-degree of the sink and network congestion. The larger the number of adjacent (gateway) nodes, the higher the level of congestion and associated retransmissions near the sink. Thus, it may be in the best interest of both the sink and the network to limit the number of gateways it associates with. However, if the number of gateways is too small, they will become overloaded by the network traffic being routed to the sink. Hence, it may be valuable to limit the *betweenness* of the gateway nodes (i.e., the number of multi-hop routes passing through each node) [10]. Our framework is capable of accounting for such constraints.

7 Related Work

The literature is rich with work focused on extending the lifetime of sensor networks [13, 14, 17, 24]. Here we present an overview of related research and identify the novelty of our contributions (summary in Table 2).

Table 2. Comparison of Related Schemes for Managing Mobile Sinks

Algorithm	Algorithm execution	Metric	Platform mobility	Metric generation
Joint Mobility and Routing [11]	Offline	Load distribution	Low	Simulated Model
Greedy Maximum Residual Energy (GMRE) [3]	Online	Residual energy	Variable	Simulated Model
Adaptive Sink Mobility [19]	Online	Data events	Variable	Observed
Deterministic/Random Walk Models [5]	None	N/A	Pre-defined	N/A
Our Dynamic Mobility Management Framework	*Online*	*Parameterizable: Residual energy, network congestion, distance to sink*	*Variable depending on network conditions*	*Observed*

Mathematical Sink Trajectory Models. Luo et al. [11] show, using a mathematical model, that when nodes are distributed according to a Poisson distribution within a circle, the (near-) optimal mobility strategy is for the sink to travel along the periphery of the circle. They argue that using such a path will improve lifetime by approximately 500% over using a static sink. They also propose an algorithm for routing to the sink. As the sink moves along the network boundary, message flows from the network "follow" the sink.

Wang et al. [21] present an LP formulation for determining an optimal sink trajectory parameterized by *sojourn time*. Their model makes several simplifying assumptions, limiting its use. Basagni et al. [3] improve upon the LP model presented in [21] by lifting some of these assumptions. They also present a new path planning scheme, *Greedy Maximum Residual Energy*: rather than pre-computing the sojourn times at different nodes in the network, the mobile sink greedily selects a neighboring node as its new location based on residual energy.

Efficient Message Routing Protocols. Baruah et al. [2] present an approach to maintaining node-to-sink routes within a network. They do not address the problem of determining an optimal mobility pattern, instead focusing on how to maintain usable data routes. Urgaonkar et al. [18] present an approach that allows static nodes to learn a sink's mobility pattern. Some nodes (*"moles"*) statistically characterize the sink's (random) movements using a probability distribution function; the result is used to inform message forwarding decisions.

Luo et al. [12] extend MintRoute [22] by adding steps to account for (*i*) link breakage, (*ii*) topological changes, and (*iii*) packet loss. They test their routing protocol using TOSSIM and illustrate the energy savings associated with using a mobile, time-synchronized sink. Chakrabarti et al. [4] use a pre-defined path to bring the mobile sink close to each node in the network to minimize long-range radio transmissions, thus reducing transmission energy.

Data Ferrying/Relaying. Jea et al. [8] present a load-balanced data collection algorithm that uses multiple mobile elements (*"mules"*) to collect data. The authors provide evidence supporting the use of large storage buffers on both the static nodes and mules to ensure data integrity. Shah et al. [15] provide insight into buffer requirements for nodes and mules within a sensor network and the effects of various buffer sizes on transmission success rates.

Somasundara et al. [1] and Kansal et al. [9] prove that mobile-sink-based sensor networks transmit fewer packets of data when compared to their static counterparts. They propose a model for calculating a data-complete trajectory for a mobile sink. Wang et al. [20] discuss two alternatives to using a mobile sink for sensor network lifetime extension in cases where utilizing a mobile sink might be infeasible. The first method uses extra static nodes near the sink. These extra nodes act like *"sleeper"* nodes within the network, and sleep for long durations, but once other nodes around the gateway begin to fail, they wake up to continue servicing the network. The second method uses additional resource-rich nodes near the sink, which together provide a rudimentary load balancing service by distributing the workload of the other gateway nodes.

8 Conclusion

We presented the design of an extensible, generalized framework for managing the trajectory of a mobile sink within a static sensor network. The framework supports objective-specific mobility decision metrics, based on which the path of a mobile sink can be computed dynamically. The framework does not require any a priori information about the sensor network, except for the extent of the area it covers. The framework periodically calculates the *quality* of each region within the network using dynamic measurements, based on an optimization-specific notion of *quality* (e.g., maximum residual node energy). This global portrait is used to drive sink mobility decisions without any assumptions of constancy or uniformity in radio reach or power consumption.

We also presented a reference implementation of the framework for TinyOS, with an emphasis on decision metrics aimed at ensuring uniform energy consumption across devices. The goal was to maximize network longevity. We evaluated this implementation using experiments on a testbed of TelosB motes. The dynamic base-station path computed by our mobility manager achieved a lower mean and standard deviation in energy consumption (1.43J and 0.13J) compared to an arbitrary path (1.56J and 0.33J), and a path computed using a typical LP formulation (1.49J and 0.23J). This means that the rate of consumption was more uniform, and consequently, the effective lifetime of the network was longer. Although the energy savings look modest, the lifetime extension was substantial.

Acknowledgments. This work was supported in part by the National Science Foundation (CNS-0746632, CNS-0745846). The authors gratefully acknowledge the NSF for its support.

References

1. Somasundara, A., et al.: Controllably mobile infrastructure for low energy embedded networks. IEEE Trans. on Mobile Computing 05(8), 958–973 (2006)
2. Baruah, P., et al.: Learning-enforced time domain routing to mobile sinks in wireless sensor fields. In: LCN 2004, Washington, DC, USA, pp. 525–532. IEEE, Los Alamitos (2004)
3. Basagni, S., et al.: Controlled sink mobility for prolonging wireless sensor networks lifetime. Wireless Networks 14(6), 831–858 (2008)
4. Chakrabarti, et al.: Communication power optimization in a sensor network with a path-constrained mobile observer. ACM TOSN 2(3), 297–324 (2006)
5. Chatzigiannakis, I., et al.: Sink mobility protocols for data collection in wireless sensor networks. In: MobiWac 2006, pp. 52–59. ACM, New York (2006)
6. Dalton, A.R., et al.: Reducing the impact of link quality variation in embedded wireless networks. Int. J. of Ad Hoc & Sensor Networks (AHSWN)
7. Ekici, E., Gu, Y., Bozdag, D.: Mobility-based communication in wireless sensor networks. IEEE Communications Magazine 44(6), 56–62 (2006)
8. Jea, D., et al.: Multiple controlled mobile elements (data mules) for data collection in sensor networks. In: Prasanna, V.K., Iyengar, S.S., Spirakis, P.G., Welsh, M. (eds.) DCOSS 2005. LNCS, vol. 3560, pp. 244–257. Springer, Heidelberg (2005)
9. Kansal, A., et al.: Intelligent fluid infrastructure for embedded networks. In: MobiSys 2004, pp. 111–124. ACM, New York (2004)
10. Freeman, L.C., et al.: Centrality in Valued Graphs: A Measure of Betweeness Based on Network Flow. Social Networks 13(141), 141–154 (1991)
11. Luo, J., Hubaux, J.-P.: Joint mobility and routing for lifetime elongation in wireless sensor networks. In: INFOCOM 2005, New York, pp. 1735–1746 (2005)
12. Luo, J., et al.: Mobiroute: Routing towards a mobile sink for improving lifetime in sensor networks. In: Gibbons, P.B., Abdelzaher, T., Aspnes, J., Rao, R. (eds.) DCOSS 2006. LNCS, vol. 4026, pp. 480–497. Springer, Heidelberg (2006)
13. Papadimitriou, I., Georgiadis, L.: Maximum lifetime routing to mobile sink in wireless sensor networks. In: Proc. IEEE SoftCOM, New York (September 2005)
14. Jain, S., et al.: Exploiting mobility for energy efficient data collection in wireless sensor networks. Mob. Netw. Appl. 11(3), 327–339 (2006)
15. Shah, R., et al.: Data mules: modeling a three-tier architecture for sparse sensor networks. In: WSNA 2003, New York, May 11, pp. 30–41. IEEE Press, Los Alamitos (2003)
16. Srinivasan, K., Levis, P.: RSSI is under appreciated. In: EmNets 2006, Boston, MA (May 2006)

17. Tong, L., Zhao, Q., Adireddy, S.: Sensor networks with mobile agents. In: MILCOM 2003, New York, pp. 688–693. IEEE, Los Alamitos (2003)
18. Urgaonkar, R., Krishnamachari, B.: FLOW: An efficient forwarding scheme to mobile sink in wireless sensor networks. In: SECON 2004, Washington, DC (2004)
19. Vincze, Z., et al.: Adaptive sink mobility in event-driven multi-hop wireless sensor networks. In: InterSense 2006, p. 13. ACM, New York (2006)
20. Wang, W., et al.: Using mobile relays to prolong the lifetime of wireless sensor networks. In: MobiCom 2005, pp. 270–283. ACM, New York (2005)
21. Wang, Z.M., et al.: Exploiting sink mobility for maximizing sensor networks lifetime. In: HICSS 2005, Washington, DC, USA, p. 287.1. IEEE Computer Society, Los Alamitos (2005)
22. Woo, A., et al.: Taming the underlying challenges of reliable multihop routing in sensor networks. In: SenSys 2003, pp. 14–27. ACM, New York (2003)
23. Younis, O., Fahmy, S.: An experimental study of routing and data aggregation in sensor networks. In: LOCAN 2005, Washington, DC, USA (November 2005)
24. Zhao, W., et al.: A message ferrying approach for data delivery in sparse mobile ad hoc networks. In: MobiHoc 2004, pp. 187–198. ACM, New York (2004)

Performance Evaluation of Network Coding and Packet Skipping in IEEE 802.15.4-Based Real-Time Wireless Sensor Networks

Marc Aoun[1], Antonios Argyriou[2,3], and Peter van der Stok[1]

[1] Philips Research, High Tech Campus 34, 5656AE, Eindhoven, The Netherlands
{marc.aoun,peter.van.der.stok}@philips.com
[2] Department of Electrical and Computer Engineering,
University of Thessaly, Greece
[3] Center for Research and Technology Hellas (CERTH), Thessaloniki, Greece
anargyr@gmail.com

Abstract. In a number of application domains, the volatility of the monitored environment where Wireless Sensor Networks (WSNs) operate engenders timing constraints on the generation, processing, and communication of sensory data. In this paper, we primarily focus on the use of an information theoretic approach, namely network coding, to improve the on-time delivery of messages in IEEE 802.15.4-compliant networks. We further study the real-time gain that packet skipping can provide on its own and in combination with network coding. Subsequently, we investigate the potential benefits of introducing deadline-awareness into the coding mechanism through the use of an Earliest Deadline First (EDF) - based scheduling policy. Simulation results show that network coding and packet skipping are by themselves effective techniques to increase on-time goodput, with additional gain obtained when the two techniques are used concurrently. Our results further provide insight into the performance of deadline-aware scheduling in a network coding environment.

1 Introduction

The advent of wireless sensor networking is shaping a future where Ubiquitous Computing and Ambient Intelligence become a reality. A key characteristic that differentiates Wireless Sensor Networks (WSNs) from other systems is their proximity to the environment they monitor. The main function of sensor networks is to report the state of such a physical environment. The latter often being dynamic and volatile means that the *state snapshot* made through sensing by a deployed WSN remains valid for a limited amount of time, after which it becomes outdated and its underlying sensory data obsolete. For example, the vital signs defining the status of a patient are dynamic by nature. Related sensed values, measured by a Body Sensor Network (BSN), would provide the state of the patient at a particular point in time. They remain valid, as well as any conclusion based on them, only as long as the vital signs are stable. Any subsequent actuation (e.g. automatic update in drug administration) is correct as long as the information it is based upon is still a valid depiction of the patient's situation. Additionally, it is expected that originally decoupled, application-specific networks, will cooperate and converge towards supporting multiple applications in a concurrent manner. Such a convergence would practically mean that different data rates and deadline requirements will have to be dealt with by the same device. Hence the importance of studying real-time aspects under different load conditions.

Given the above emerging scenarios, we take a look at WSNs from a real-time perspective where the definition of correctness now spans two dimensions: correctness in computation/sensing/analysis, and meeting deadlines when delivering information and performing actions. At the communication level, two aspects come into light: the timely delivery of information to achieve correct application behavior (e.g. timely and correct actuation) under different network loads, and the elimination of obsolete information in order to avoid false conclusions and reduce useless overhead.

In this paper we investigate a novel approach for improving real-time performance in sensor networks by considering recent information-theoretic advances. More specifically, we explore the idea of network coding [1] where routing elements in a network execute algebraic coding operations on packets besides simply forwarding them. Network coding increases the information content per packet transmission without incurring significant overhead [13]. In this paper we leverage this advantage that network coding provides in order to expedite the transmission of real-time packets through the network. Although we explore the above characteristic in a simple network topology, the presented results are transferable to more complex networks. The reason is that in generic multi-hop topologies, router nodes are the meeting point of packets coming from different neighboring nodes and are responsible of forwarding them towards their destinations. A router node can take advantage of the communication pattern between its neighbors and the broadcast nature of the wireless channel; instead of forwarding each packet separately, the router can opportunistically combine two packets through an algebraic XOR operation, and broadcast one resulting coded packet instead of two packets. Provided that nodes store copies of packets they have recently transmitted, the two concerned direct neighbors of the router can decode (through a XOR operation) the coded packet and retrieve the information content relevant to them. Fig. 1(a) demonstrates the basic concept, where node A wants to communicate packet 1 to node B, and C is communicating packet 2 to node D, with node R being a common routing point for both flows. The following communication sequence takes place: Node A broadcasts packet 1 and C broadcasts packet 2. nodes M and N rebroadcast packet 2 and packet 1, respectively. Node R codes the two packets (1 XOR 2) and broadcasts the resulting coded packet. Node M retrieves packet 1 by decoding the received packet ([1 XOR 2] XOR 2) and forwards it to node B. Node N performs a similar decoding operation ([1 XOR 2] XOR 1) and delivers packet 2 to node D.

Our contribution in this paper is multi-fold: 1) We investigate network coding from a real-time perspective, i.e. by assessing its impact on the delivery of

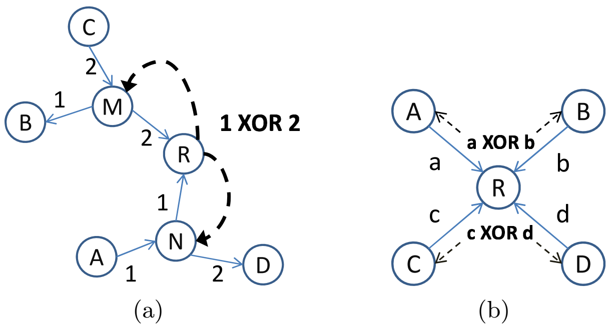


Fig. 1. Network coding of two and four packet flows

real-time data (goodput) that are associated with *per-packet deadlines*. This is in contrast to previous non-sensor networking work where coding was primarily employed with the purpose of improving throughput and delay optimization at the flow-level. 2) We investigate the impact that packet skipping (i.e. proactive message removal from transmission queues) can have on timeliness. 3) We investigate the joint real-time performance of the combination of coding and packet skipping, and assess their relative contributions for different loads and different timing requirements. 4) We provide insight into the performance of deadline-aware packet scheduling in a network coding environment, relative to the widely adopted and simpler to implement First Come First Served (FCFS) scheduling policy.

To the best of our knowledge, our work is the first to consider network coding over IEEE 802.15.4 networks and to investigate coding as a method to improve real-time performance. We developed a performance model that characterizes analytically the real-time performance of a single M/M/1 node with network coding in [2]. The complexities we faced in that work forced us to seek a simulation-based study that we present here.

2 Related Work

A fair amount of work has targeted real-time aspects in WSNs. Existing approaches try to address timeliness at specific layers separately [6,9,8,10,12], or with sophisticated cross-layer techniques [7,15]. Caccamo et al. designed an implicit prioritized protocol based on the Earliest Deadline First (EDF) Policy for hard real-time sensor networking [6]. A cellular topology is adopted and channel diversity is used among neighboring cells. Intra-cell communication is achieved by replicating the transmission schedule at all cell nodes and employing EDF to schedule channel access. Inter-cell communication uses a globally synchronized TDMA mechanism. He et al. proposed SPEED [10], a protocol that provides real-time unicast, area-multicast and area-anycast services. It achieves soft real-time end-to-end communication by maintaining a certain velocity for a packet travelling across the network. RAP, a cross-layer real-time communication architecture designed by Lu et al. [15], provides high-level services to applications and introduces the concept of Velocity Monotonic Scheduling (VMS), a packet scheduling policy that takes into account both the deadline of packets and the remaining distance they have to cover to reach their destination. More recently, Aoun and van der Stok [3] provided an in-depth schedulability analysis of real-time periodic messages in an overloaded point-to-point IEEE 802.15.4 setting, and analyzed the performance of different service rejection criteria, identifying the optimal strategies for different operational conditions. The idea of network coding has been applied in IEEE 802.11-based multi-hop wireless networks and throughput benefits in the order of 3-4 times over baseline 802.11 have been demonstrated for bulk data transfers [13,14,4]. In this paper we take an additional step by studying its performance in the case where packets have specific delivery deadlines, i.e. when throughput and goodput hold different meanings.

3 System Model

The system under consideration is an IEEE 802.15.4 mesh network and is depicted in Fig. 1(b). The four nodes A, B, C, and D, referred to as source nodes, communicate with each other in pairs, acting concurrently as sources and sinks of information. The nodes are not within reliable communication range, thus requiring a router. This scenario is realistic since the transmission range is usually quite shorter than the carrier sensing range [17]. Our topology choice of having the sources as direct neighbors of the router R is motivated by a need to reduce the complexity of the analysis, while maintaining representative results and shedding light on the fundamental performance of network coding.

Source nodes maintain two queues; a transmission queue (TxQ) and a decoding queue (DcQ). The TxQ holds messages that are generated by the node and need to be transmitted. The DcQ holds copies of messages that have already been transmitted. These copies are used to decode algebraically coded packets and retrieve their information content. The router node maintains n reception queues RxQ_i. A reception queue RxQ_i is associated with source node i and functions as forwarding queue; it stores messages that were received from that source node and need to be forwarded to its counterpart node. All the aforementioned queues are implemented at the network layer. No modifications are introduced to the IEEE 802.15.4 MAC.

3.1 IEEE 802.15.4

The IEEE 802.15.4 MAC/PHY standard [11] is the most popular standard for low data rate wireless PANs. The standard specifies four physical layers (PHY) with three of them being based on direct sequence spread spectrum (DSSS) techniques. We consider the DSSS PHY working in the 2450 MHz band with a data rate of 250 Kbps. Regarding the MAC layer, the IEEE 802.15.4 specifies a beacon-enabled mode and a beaconless mode. In beacon-enabled mode, channel access is arbitrated through the use of a slotted CSMA/CA algorithm. Beaconless mode employs non-slotted CSMA/CA. In this work we adopt the beaconless mode.

Three configurable attributes define the functioning of non-slotted CSMA/CA Medium Access Control: *macMinBE*, *macMaxBE*, and *macMaxCSMABackoffs*. The algorithm functions as follows: When a node has a packet to send, it first chooses a random number n of backoff periods (each equal to 320μs), where n is uniformly distributed between 0 and 2^{BE}-1. The variable BE refers to the Backoff Exponent, that is initially set to *macMinBE*. Once n backoff periods elapse, the node checks whether the channel is free or not. If it is free, the node initiates the packet transmission. If on the other hand the channel is busy, BE is incremented by 1, unless it has already reached the value adopted by *macMaxBE*. In that latter case, BE is set to *macMaxBE*. The process of choosing a random number n of periods, waiting for them to elapse and then checking the channel is again repeated. In total, the MAC will try this process *macMaxCSMABackoffs* times, after which it will report a failure to the upper layer in case the channel is never found to be free. Otherwise, a success will be reported (we assume broadcast transmissions without ACKs). As a consequence of the CSMA/CA mechanism, the time it takes to service a packet is variable.

4 Coding, Scheduling, and Skipping Mechanisms

Three mechanisms were implemented on top of the 802.15.4 MAC layer and are described below. Note that all these mechanisms are closely related with respect to their functionality and their impact on performance. In this paper we assume that new packets with firm deadlines are generated at the application layer of the source nodes. A generated packet is forwarded to the network layer and is subsequently presented for admission at the *TxQ*. If the queue is full, the packet is discarded (Drop-Tail policy). Otherwise, it is admitted. Similar admission behavior takes place at the router node for the *RxQ*s.

4.1 Packet Coding

Digital network coding, when enabled, occurs at the router node when both RxQs belonging to a pair of inter-communicating source nodes are populated, and the MAC layer is ready to service a packet. The payloads of the Head-of-Line (HOL) packets of the two queues are algebraically coded with each other, resulting in a coded packet. The sequence number of both packets and the IDs of the two source nodes are appended to the payload, resulting in a minor extra overhead of 4 bytes. Similarly to the Sliding Window Protocol, we assume periodic reuse of the available range of sequence numbers. In case only one RxQ is populated, its HOL packet is given to the MAC for servicing. In that sense, network coding is opportunistic in that it exploits the coincidental presence of two packets that can be coded. In essence, a synchronous presence is not required per se, but is leveraged when it occurs.

When a coded packet is received at a source node, the latter will first check whether it is one of the two intended recipients. It will subsequently check the appended sequence numbers and search its DcQ for a copy of its native packet used in the coding process. In case a copy is found, the payloads of the coded packet and the copy are XORed with each other, resulting in the payload intended for the source node. Otherwise, the decoding process fails and the packet is considered as lost.

4.2 Scheduling and Queue Management

The TxQs at the source nodes are served using the First Come First Served (FCFS) scheduling policy, where the packet with the smallest queue admission time among all packets residing in the queue is committed first to the MAC layer for servicing. The scheduling of packets at the router operates as follows: 1) A "horizontal" FCFS mechanism orders the packets in each RxQ in increasing order of queue admission time, 2) A "vertical" FCFS mechanism decides which RxQ (two of them when network coding is enabled) to serve next. This "vertical" FCFS is greedy: It does not consider a combination of admission times for two packets that can be coded. It bases its scheduling decision on one admission time (the smallest). In case no coding opportunity exists, the native FCFS chosen packet is passed directly to the MAC. The concept of the two-dimensional FCFS scheduling that we just described is illustrated in Fig. 2. To assess the impact of deadline-awareness in the coding process, we shall also investigate at a later stage in this paper the performance of an Earliest Deadline First (EDF)-based scheduling policy (both horizontal and vertical) at the router node.

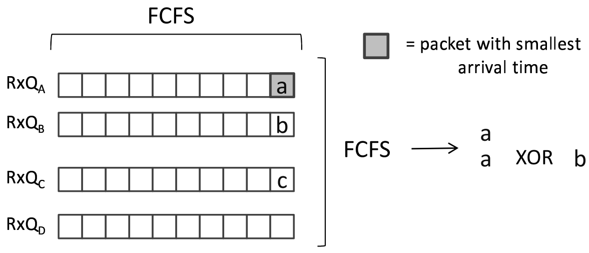


Fig. 2. Two-dimensional scheduling at the router node

The proposed scheduling and coding algorithms, as well as all transmission queues, are implemented at the network layer. No queue build-up happens at the MAC: At all nodes, packets are committed one at a time to the MAC thus allowing full decision control at the network layer over the precise packets that will be served and the packets that will be denied service (skipped). At a source node, a copy of a transmitted packet is stored in the *DcQ*. The copy is removed when the source node overhears R forwarding this packet (in native or coded form) or any other previously committed packet that has: 1) a higher sequence number (in case of FCFS at the router), or 2) a higher sequence number and a bigger absolute deadline (in case EDF is adopted at the router). When the DcQ is full, no copy is stored and the packet is flagged to inform node R that it should not be coded.

4.3 Packet Skipping

Packet skipping refers to the action of denying future service (i.e. transmission) to a packet that is already residing in a transmission queue. The procedure, when activated, proactively cleans the queues upon the arrival of new packets in order to potentially reserve the buffer space for packets that have higher probability of arriving on-time. Packet skipping is executed before coding and before any type of packet is committed to the MAC layer in order to avoid coding and serving packets that are expired or are unlikely to meet their timing constraints.

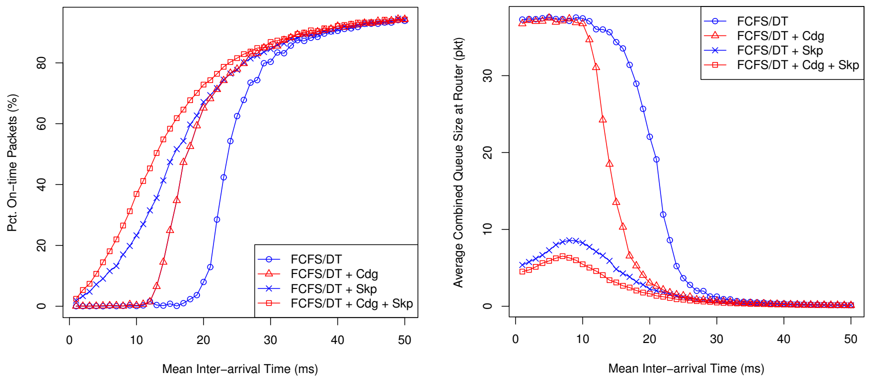
To decide whether to execute packet skipping for a particular packet, an estimate of the MAC service time is used. Estimation is needed due to the variable service time mentioned in section 3.1. The estimate is set to the mean of the service time values of the last 10 serviced packets. To account for the actual 2-hop communication path, source nodes multiply the calculated estimate by 2. Thus, they rely on a rather optimistic estimation of the waiting time plus service time that their packets will experience at the router node. The lead time (i.e. the remaining time until deadline expiration) of every packet in the queue is compared to the estimated service time. If the lead time is smaller, the packet is dropped from the queue. Otherwise, it is maintained.

5 Performance Evaluation

5.1 Simulation Setup

We have built our performance assessment environment in OMNet++ [18], on top of simulation models of wireless propagation, multiple access interference, radio state machine and the IEEE 802.15.4 non beacon-enabled MAC protocol. The accuracy of the models for timeliness evaluation has been validated by Rousselot et al. [16]. The MAC layer is configured with $macMinBE$=3, $macMaxBE$=5, and $macMaxCSMABackoffs$=4, which correspond to the default values specified in the standard [11]. The network consists of 4 source nodes [A,B,C,D] and a router R, as depicted in Fig. 1(b). The communication pattern is between A and B (flow 1), and between C and D (flow 2).The size of TxQ for each source node is equal to 10 packets and the DcQ has a size equal to 20 packets. Each of the RxQ queues at node R can hold up to 10 packets.

Source nodes generate packets following an exponentially distributed inter-arrival time with mean τ. The smaller τ is, the higher is the generation rate and thus the higher the network load is. Each node generates 10000 messages in total. The value of τ is gradually varied in the simulations to assess the system performance under different traffic loads. Packets have to traverse two



(a) On-time performance (b) Average buffer occupancy at node R

Fig. 3. (a) Percentage of packets received at the source nodes within their deadlines and (b) Average buffer occupancy at node R, for the baseline case, coding (Cdg), skipping (Skp) and the combination of Cdg and Skp

hops (e.g. A-R-B). Plain packets have a physical frame size of 60 bytes and coded packets 64 bytes. Every generated packet has a firm absolute deadline associated with it, after which its information content becomes obsolete. The absolute deadline is equal to the absolute generation time of the packet plus a relative deadline. In the first part of the evaluation, relative deadlines are uniformly distributed over the interval [10ms,100ms], thus with a mean value equal to 55ms and a width of 90ms. In a second phase, we shall vary the average relative deadline in order to assess the performance of coding and skipping for different timeliness requirements. In all presented figures, FCFS/DT refers to the simple baseline mechanism (First Come First Served with Drop-Tail). The acronym *Cdg* refers to network coding, and *Skp* refers to packet skipping. Four mechanisms are investigated: the baseline case (FCFS/DT), the baseline case plus network coding (FCFS/DT + Cdg), the baseline case plus packet skipping (FCFS/DT + Skp), and finally, the baseline case plus a combination of coding and skipping (FCFS/DT + Cdg + Skp).

5.2 Case 1: Constant Average Deadline

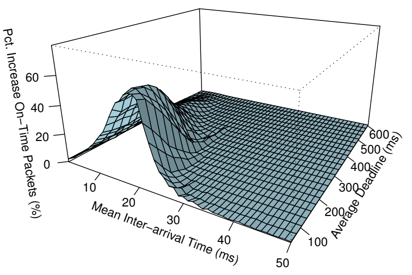
In this section, the relative deadline of generated packets is uniformly distributed between 10ms and 100ms. Fig. 3(a) presents, for all mechanisms, and as a function of τ, the percentage of packets that make it on time to their destination out of all 40000 packets generated in the network. Fig. 3(b) provides the average buffer space occupancy at the router node (total over all RxQs). We recall that the larger τ is, the lighter the network load is.

The results show that the timeliness performance of baseline FCFS/DT is the worst among the four mechanisms, in particular for high loads (1ms $< \tau <$ 30ms). For larger τ values, all methods have similar performance, since the network load is low and the queueing delays are small. Applying algebraic coding at node R provides a gain that reaches up to 57% over the baseline case. This performance improvement can be explained by studying Fig. 3(b). By XOR-ing payloads and servicing two packets concurrently, network coding considerably reduces the overall buffer usage at node R, relative to baseline FCFS/DT. The drop in buffer occupancy is visible in the same operation region where the on-time gain is noticed (10ms $< \tau <$ 30ms). A reduction in buffer space occupancy reduces the number of messages dropped at node R by the Drop-Tail policy. The ability of FCFS/DT + Cdg to service two packets concurrently further reduces the queueing time that packets have to endure at node R before having access to the MAC layer for servicing.

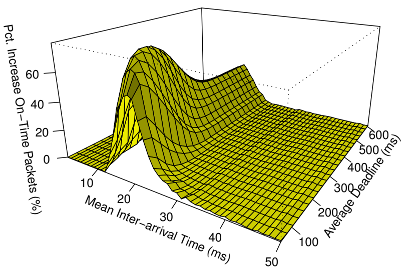
The relative goodput gain is nevertheless negligible for values of τ smaller than 10ms. There are two reasons behind this: 1) For such small inter-arrival times, a bottleneck exists at the source nodes, with a high percentage of packets being dropped at these nodes by the Drop-Tail mechanism or otherwise having to experience significant waiting times in the TxQ queues once admitted (the average queue size at the source nodes was measured to be equal to 9.5 packets) 2) Due to the high packet generation rate, The DcQ at the source nodes become considerably loaded, with a significant amount of sent packets being flagged as not valid for coding. This flagging process reduces the ability of node R to code packets, and its impact is reflected in the average buffer occupancy; as shown in Fig. 3(b), no difference in buffer occupancy between FCFS/DT and FCFS/DT + Cdg is noted for τ smaller than 10ms. When assuming a DcQ of infinite size, a drop of 25% in this same region was noticed.

Packet skipping outperforms both baseline FCFS/DT and simple algebraic coding. By proactively dropping packets that have a high probability of missing their deadlines, skipping favors those packets which are more likely to satisfy their timing requirements. On the other hand, simple FCFS/DT and its network coding variant FCFS/DT + Cdg treat every single packet that is admitted to the transmission queue. Both remain oblivious to the fact that buffering and servicing expired packets and packets with short lead times consumes both precious buffer space and MAC service time. This in turn increases the number of packets rejected by Drop-Tail and increases the waiting time of other admitted packets with more relaxed deadlines. The reduction in congestion obtained through packet skipping is correlated with a reduction in buffer occupancy, as illustrated in Fig. 3(b). A similar reduction in buffer occupancy at the source nodes was also observed.

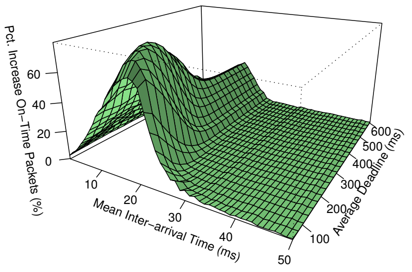
The ability of network coding to simultaneously serve two packets at router R, adds to the efficient buffer space and MAC usage provided by packet skipping. As shown in Fig. 3(a), a combination of coding and skipping performs the best among all tested cases. Applying network coding in addition to proactively cleaning queues reduces the waiting time of packets at node R and reduces further the average queue size at the router (Fig. 3(b)). Overall, applying a combination of network coding and packet skipping can provide up to 65% more on-time packets than traditional FCFS Drop-Tail. Finally, it is also important to mention



(a) FCFS/DT + Skp



(b) FCFS/DT + Cdg



(c) FCFS/DT + Cdg + Skp

Fig. 4. Percentage difference in on-time packets for coding, skipping, and their combination, relative to baseline FCFS/DT

that the higher performance of both skipping and coding is also coupled with a decrease in transmitted messages, and thus comes at less transmission effort.

5.3 Case 2: Varying Average Deadlines and Packet Arrival Rates

In this section we investigate the real-time performance of coding, skipping, and their combination, for different timeliness requirements and different loads. For that purpose, the average relative deadline of generated packets is varied from 55ms to 605ms in steps of 25ms, while maintaining a uniform deadline distribution with a constant width of 90ms. The network load is also varied by varying the mean inter-arrival time of packets at the source nodes from 1ms to 50ms in unit steps. The difference in on-time packets between each of the three methods and baseline FCFS/DT is plotted in Fig. 4. Note that the percentage values are relative to the total 40000 generated packets.

We shall first consider the impact of solely applying packet skipping (i.e. FCFS/DT + Skp). Fig. 4(a) shows a clear performance benefit of skipping for mean inter-arrival times τ between 1ms and an upper limit τ_l. The value τ_l for which performance improvement is still witnessed decreases when the average deadline is increased; the higher the deadline is, the narrower is the range of τ values for which packet skipping outperforms baseline FCFS/DT. Indeed, higher deadlines implies that less packets will get expired or be assessed as outdated, which further implies that less packets will be denied service. The peak increase, that initially reached up to 60% for an average deadline of 55ms, gradually fades away for higher deadlines values, with less than 20% peak improvement left at a deadline of 350ms, and 5% for a deadline of 500ms.

Moving to the case of FCFS/DT + Cdg, a number of performance-related aspects can be deduced from Fig. 4(b). The most striking difference between FCFS/DT + Skp and FCFS/DT + Cdg is that, unlike skipping where the gain gradually decreases for higher deadline values, coding has the ability to maintain a constant gain for large average deadlines, with a constant peak gain of more than 30% for deadlines above 400ms. The ability of coding to reduce buffer occupancy time and thus reduce the number of dropped packets at node R provides this constant gain irrespective of the average deadline. Whereas skipping is able to provide improvement for small τ values (e.g. τ less than 10ms for a 55ms average deadline), coding alone does not deliver gain over baseline FCFS/DT under these high arrival rates, for the same reasons mentioned in section 5.2. The actual lower limit of τ above which coding starts to provide performance gain is deadline-dependent. A further aspect where coding and skipping differ is in the evolution of the peak performance relative to the deadline value; for all values of τ, skipping provides the highest improvement for the lowest considered average deadline (55ms), whereas the performance of coding first increases when the deadline is increased, before reaching a maximum value beyond which it gracefully decreases to the observed constant gain value.

To assess the combination of coding and skipping, we plot in Fig. 4(c) the percentage on-time difference for the combined FCFS/DT + Cdg + Skp approach. We also plot in Fig. 5 the gain difference between FCFS/DT + Cdg + Skp and FCFS/DT + Skp. The general characteristics mentioned for FCFS + Cdg still hold for the combined approach, with one major exception; FCFS/DT + Cdg + Skp provides gain even for the smallest range of τ. Fig. 5 shows that it is the contribution of packet skipping that accounts for this difference. Interestingly, this contribution is shifted to lower values of τ compared to the FCFS/DT + Skp case. This is explained by the fact that network coding operates in the same region as skipping, thus reducing the impact that skipping would otherwise have had if it were applied alone. The impact of skipping remains in the region where coding was not initially providing significant gain, i.e. for the highest network loads.

5.4 Peak Performance and Optimal Operation Point

The results plotted in Fig. 4 indicate that for every considered average deadline, there exists a specific value of τ for which the maximum gain is achieved. This value will be referred to as τ_{max}. We shall try to elucidate the existence of such an optimal τ using FCFS/DT + Cdg as illustrative example. Fig. 6(a) plots the measured value of τ_{max} for every average deadline. It additionally plots the value of τ, referred to as τ_p, for which the maximum increase in delivered throughput

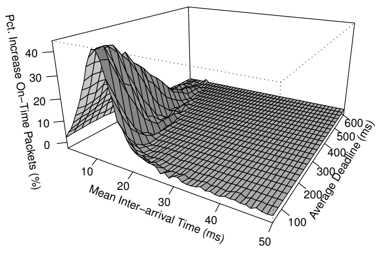
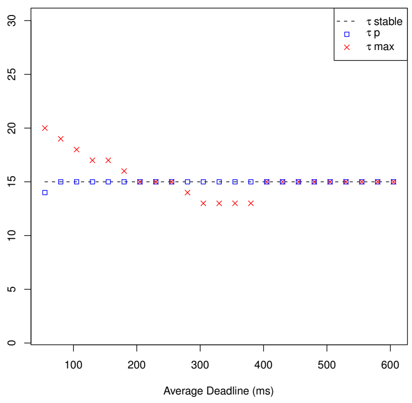


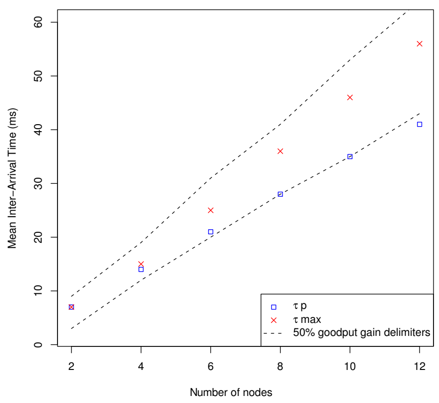
Fig. 5. On-time difference between FCFS/DT + Cdg + Skp and FCFS/DT + Skp

at the sink nodes occurs (regardless of whether the packets made it within their deadline or not). Finally, the stability point τ_{stable} of the queueing system at the router node is also plotted. Stability, from a queueing theory perspective, is the operation regime where a queueing system reaches and maintains an equilibrium state, i.e. when the arrival rate of packets at the queue is smaller or equal to the average departure rate. The stability point was identified by measuring the average packet arrival rate and average departure rate at node R, and identifying the smallest value of τ for which the ratio of these two rates is equal to 1. Note that node R has the highest load among all nodes in the network. As such, source nodes are also in a stable state at τ_{stable}.

For all deadlines, τ_p remains equal to τ_{stable}, meaning that the relative *throughput* increase is always maximized at the stability point. Indeed, the stability point offers the maximum number of opportunities for coding, while reducing the number of packets dropped by the Drop-Tail mechanism. A major observation is that for deadlines bigger or equal to 400ms, τ_{max} and τ_p are exactly equal to τ_{stable}; for high deadline values, the peak increase in on-time packets relative to FCFS/DT is achieved at the stability point of the network. On the other hand, for deadlines up to 200ms, the goodput is maximized for τ_{max} values bigger than τ_{stable} and τ_p. For such deadlines, even though the relative throughput is still maximized at τ_{stable}, a lower network load is needed to reduce the waiting time of packets in the queues. For deadlines between 300ms and 400ms, the shift between τ_{max} and τ_{stable} is due to the fact that the goodput increase is measured as a difference to the baseline case, which unlike coding, still does not provide any positive gain for this deadline range.

Finally, to present insight into the behavior of the system for higher node densities and loads, Fig. 6(b) plots, for a fixed deadline of 200ms and an increasing number of nodes: τ_p, τ_{max}, and the range of τ for which the provided goodput increase is at least 50% of the increase provided at τ_{max}. An observation we can make here is that the bigger/more loaded the network is, the more τ_{max} diverts from τ_p, and the wider the 50% range of τ values becomes. The aforementioned results validate our position of looking at network coding from a real-time goodput perspective, versus looking at it from a throughput perspective.





(a) Optimal operation point for varying deadlines.

(b) Optimal operation regime vs number of source nodes.

Fig. 6. Peak performance and optimal operation points

5.5 Deadline-Aware Coding Using EDF Scheduling

We have till now assessed the performance of network coding while maintaining the same scheduling policy (FCFS). A yet unexplored dimension is the order with which packets are serviced at the router R. The following section assesses the impact of servicing queues based on the absolute deadlines of packets. Towards that end, FCFS is replaced by an Earliest Deadline First (EDF) variant: The packet P with the smallest absolute deadline among all packets residing in the RxQs is chosen first for service. In case a packet exists in the second RxQ belonging to the same flow, this packet can be opportunistically coded with P. Two scenarios are considered: In the first scenario, the average relative deadline of both flows is increased from 55ms to 605ms. In the second scenario, one flow has a fixed average deadline of 55ms, while the average relative deadline of the second flow is varied from 55ms to 605ms. This scenario is suitable for studying the impact of the vertical scheduling component. Given the insight obtained in the previous sections, we shall restrict ourselves to a comparison between FCFS/DT + Cdg + Skp and EDF/DT + Cdg + Skp.

Fig. 7 conveys the results of the first scenario (on-time difference for EDF relative to FCFS). Little difference is observed. Only minor improvements of 1-2% exists, for very limited operation points. In fact, for τ smaller than 10ms and high deadlines such as 400ms, we observed a decrease between 2-5% for EDF, that we found to be due to a less effective DcQ cleaning in the case of EDF than in the case of FCFS (As mentioned in section 4.2, for FCFS, the sequence number of an overheard packet is enough to clean smaller sequence-numbered packets from the DcQ, whereas for EDF, both the overheard sequence number and the absolute deadline should be bigger than those of a stored packet to safely remove the latter). The obtained result might seem striking at first, given the proven optimality of EDF in real-time systems. In fact, EDF has been proven

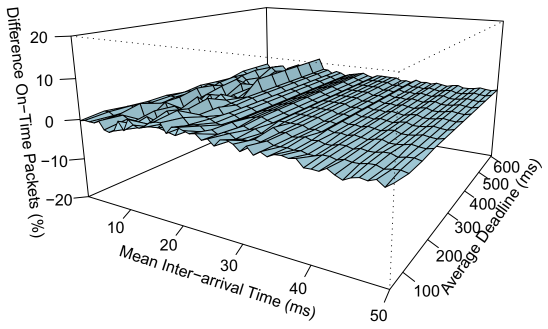
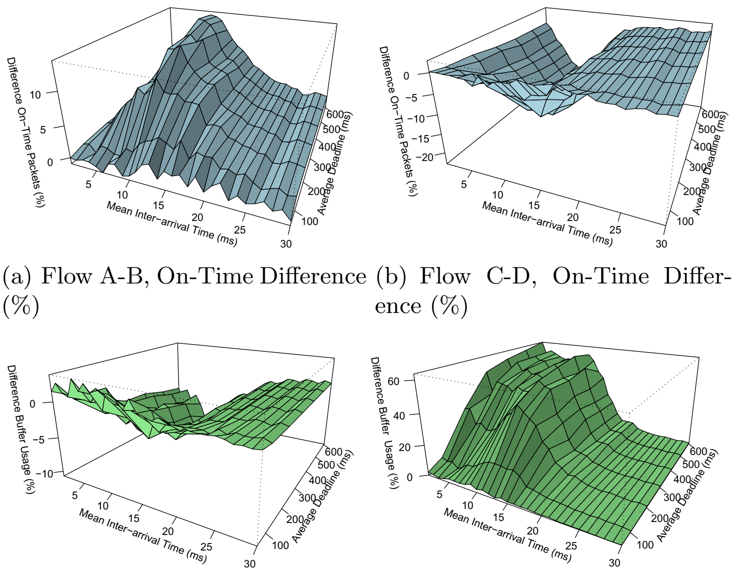


Fig. 7. Deadline-aware coding: On-time difference relative to FCFS

to be optimal *only in the case where all tasks (in our case all packets) can be serviced in an order such that all the deadlines are met*, i.e. if a feasible schedule exists [5]. This is obviously not the case here, because of the load conditions, the limited available buffer space, the stochastic service time introduced by the CSMA/CA mechanism and the transmission failures that occur.



(a) Flow A-B, On-Time Difference (%)
(b) Flow C-D, On-Time Difference (%)
(c) Flow A-B, Buffer Usage Difference(%)
(d) Flow C-D, Buffer Usage Difference(%)

Fig. 8. Per-flow on-time performance and buffer space usage difference between EDF and FCFS

Fig. 8 illustrates, for the second scenario, the per-flow on-time and buffer usage difference of EDF relative to FCFS. By taking the absolute deadlines into account, vertical EDF favors the communication flow A-B since it has, on average, smaller deadlines. This results in an improved on-time performance relative to deadline-oblivious FCFS (Fig. 8(a)) and is correlated with a decrease in the buffer space usage at the router node for the RxQs that hold packets from A and B (Fig. 8(c)). The gain in the A-B flow comes at the expense of the communication flow C-D, as shown in Fig. 8(b). Due to the favoring of flow A-B, the packets of flow C-D experience more waiting time at node R. This reflects itself in an increased buffer occupancy for this flow, as conveyed in Fig. 8(d), which in turn results in a higher dropping rate of packets by the Drop-Tail mechanism.

6 Conclusion

This work is the first to propose and investigate the use of network coding for improving the real-time performance in IEEE 802.15.4-based wireless sensor networks. Our results show that coding, packet skipping, and especially their combination, can be effective techniques that significantly increase the number of on-time received packets. The actual on-time gain depends on both the network load and the timing constraints of packets. Nevertheless, unlike packet skipping, network coding is able to maintain a constant gain for increasing deadline values. In addition to the novel approach that looks at network coding from a real-time perspective and investigates coding on top of IEEE 802.15.4 networks, this work opens the door to a number of new directions. More specifically, there is a need for analytical models that characterize the performance of packet skipping and network coding in general networks. In the domain of real-time scheduling, a comparative analysis between different scheduling algorithms in the presence of coding and skipping would be an interesting step forward in the topic, especially for varying load conditions.

Acknowledgments. Antonios Argyriou would like to acknowledge the support from the European Commission through the STREP project CONECT (FP7ICT257616).

References

1. Ahlswede, R., Cai, N., Li, S.Y.R., Yeung, R.W.: Network information flow. IEEE Transactions on Information Theory 46(4), 1204–1216 (2000)
2. Aoun, M., Beekhuizen, P., Argyriou, A.: An analytical study of network coding in the presence of real-time messages. In: IEEE International Symposium on Network Coding (NetCod), Toronto, Canada (June 2010)
3. Aoun, M., van der Stok, P.: Overloading an IEEE 802.15.4 point-to-point connection with real-time messages. In: 31st IEEE Real-Time Systems Symposium (RTSS), San Diego, CA, USA, pp. 225–235 (December 2010)

4. Argyriou, A.: Wireless network coding with improved opportunistic listening. IEEE Transactions on Wireless Communications 8(4), 2014–2023 (2009)
5. Buttazzo, G.C.: Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. Kluwer Academic Publishers, Dordrecht (2000)
6. Caccamo, M., Zhang, L.Y., Sha, L., Buttazzo, G.: An implicit prioritized access protocol for wireless sensor networks. In: 23rd IEEE Real-Time Systems Symposium (RTSS), Austin, TX, USA, pp. 39–48 (December 2002)
7. 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: 14th IEEE International Workshop on Quality of Service, New Haven, CT, USA, pp. 83–92 (June 2006)
8. Felemban, E., Lee, C.G., Ekici, E.: MMSPEED: Multipath multi-speed protocol for QoS guarantee of reliability and timeliness in wireless sensor networks. IEEE Transactions on Mobile Computing 5(6), 738–754 (2006)
9. Francomme, J., Mercier, G., Val, T.: A simple method for guaranteed deadline of periodic messages in 802.15.4 cluster cells for control automation applications. In: 11th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, Prague, Czech Republic, pp. 270–277 (December 2006)
10. He, T., Stankovic, J., Lu, C., Abdelzaher, T.: SPEED: a stateless protocol for real-time communication in sensor networks. In: 23rd International Conference on Distributed Computing Systems, Providence, RI, USA, pp. 46–55 (May 2003)
11. IEEE Std 802.15.4-2996, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), IEEE (September 2006)
12. Karenos, K., Kalogeraki, V.: Real-time traffic management in sensor networks. In: 27th IEEE Real-Time Systems Symposium, Rio de Janeiro, Brazil, pp. 422–434 (December 2006)
13. Katti, S., Katabi, D., Hu, W., Rahul, H., Medard, M.: The importance of being opportunistic: Practical network coding for wireless environments. In: Allerton Conference, Monticello, IL, USA (September 2005)
14. Katti, S., Rahul, H., Hu, W., Katabi, D., Medard, M., Crowcroft, J.: XORs in the air: Practical wireless network coding. In: SIGCOMM, Pisa, Italy (September 2006)
15. Lu, C., Blum, B., Abdelzaher, T., Stankovic, J., He, T.: RAP: A real-time communication architecture for large-scale wireless sensor networks. In: 8th IEEE Real-Time and Embedded Technology and Applications Symposium, San Jose, CA, USA, pp. 55–66 (September 2002)
16. Rousselot, J., Decotignie, J.D., Aoun, M., van der Stok, P., Oliver, R.S., Fohler, G.: Accurate timeliness simulations for real-time wireless sensor networks. In: 3rd UKSim European Symposium on Computer Modeling and Simulation, Athens, Greece (November 2009)
17. Tse, D., Viswanath, P.: Fundamentals of Wireless Communication. Cambridge University Press, Cambridge (2005)
18. Varga, A.: The OMNeT++ discrete event simulation system. In: 15th European Simulation Multiconference (ESM 2001), Prague, Czech Republic (June 2001)

Opportunistic Packet Scheduling in Body Area Networks

K. Shashi Prabh[1] and Jan-Hinrich Hauer[2]

[1] Center for Real-Time Systems Research (CISTER),
School of Engineering (ISEP),
Polytechnic Institute of Porto, Portugal
ksp@isep.ipp.pt

[2] Telecommunication Networks Group
Technische Universität Berlin, Germany
hauer@tkn.tu-berlin.de

Abstract. Significant research efforts are being devoted to Body Area Networks (BAN) due to their potential for revolutionizing healthcare practices. Energy-efficiency and communication reliability are critically important for these networks. In an experimental study with three different mote platforms, we show that changes in human body shadowing as well as those in the relative distance and orientation of nodes caused by the common human body movements can result in significant fluctuations in the received signal strength within a BAN. Furthermore, regular movements, such as walking, typically manifest in approximately periodic variations in signal strength. We present an algorithm that predicts the signal strength peaks and evaluate it on real-world data. We present the design of an opportunistic MAC protocol, named BANMAC, that takes advantage of the periodic fluctuations of the signal strength to achieve high reliability even with low transmission power.

1 Introduction

As the fraction of the aging population is increasing, the load on the healthcare services is also growing. Yet, there is a severe current and projected shortage of healthcare personnel. For example, a shortage of 1 million registered nurses by the year 2020 is projected within the USA alone [26]. Networks of sensors around as well as inside the human body, referred to as Body Area Networks (BAN), promise to revolutionize healthcare practices as they facilitate, among other things, better diagnosis, fast emergency response and personalized medication [12]. But although BANs have the potential to enable low-cost, personalized healthcare systems, it is still unclear whether they can meet the stringent QoS requirements imposed by some applications. To limit the interference to neighboring BANs and to keep the specific absorption rate (SAR) as low as possible in the interest of protecting the human tissues, it is desirable that the transmission power be kept low. The error-proneness of the low-power wireless communication, however, is a major challenge. Although the distances between devices in BANs are usually small, the wireless signal may experience severe attenuation from human body shadowing [17]. Furthermore, the changes in the environment as well as relative distance and orientation of the devices resulting from human mobility can introduce significant variations in the quality of the wireless signal.

We begin this paper with an experimental investigation of the signal strength dynamics within a BAN during periodic human movements, such as walking. We observe that the periodic changes in the relative positions of the limbs typically manifest in significant periodic changes in received signal strength (of several dBs). Furthermore, the signal strength amplitudes are typically long-lasting, of the order of 100s of milliseconds, when compared to the airtime of packets, which are of the order of only a few milliseconds. To exploit this effect we propose BANMAC, a MAC protocol that is built on the idea of opportune packet transmission, that is, packets are scheduled such that they are transmitted when signal strength is high, because then the chances are better that the packets are received correctly.

The main contributions of this paper are:

1. An empirical characterization of RSSI fluctuations in a BAN while the subject is walking *outdoors*. This work is complimentary to some of the studies done by the IEEE 802.15.6 working group in *indoors* settings [6]. Contrary to the indoors study [6], we did not always find significant differences in RSSI measurements due to internal versus external antenna. Specifically, the node placements where body shadowing is significant resulted in no significant difference.
2. The design and evaluation of an RSSI-based opportune transmission windows prediction algorithm.
3. BANMAC, a MAC protocol for BANs that schedules transmissions opportunistically when the link margin is likely to be higher than the average. We have designed BANMAC to be compatible with the recommendations of the IEEE 802.15.6 working group for MAC protocols in BANs.

The rest of this paper is organized as follows: in Sec. 2 we present a description of our experimental setups and report on typical signal strength fluctuations that we observed in BANs. We present the details and evaluations of an opportune transmission window prediction algorithm in Sec. 3. We then present BANMAC that is based on the idea of transmitting during high RSSI windows in Sec. 4. We discuss related work in Sec. 5 and conclude the paper in Sec. 6.

2 RSSI Measurements

This section describes our experimental setup and reports on typical RSSI fluctuations observable in a BAN when the subject is walking. We performed experiments with three different node platforms: Shimmer2, TelosB and MicaZ.

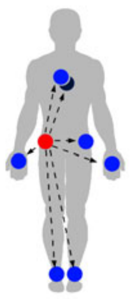


Fig. 1. Node positions: the sender is in the right pocket

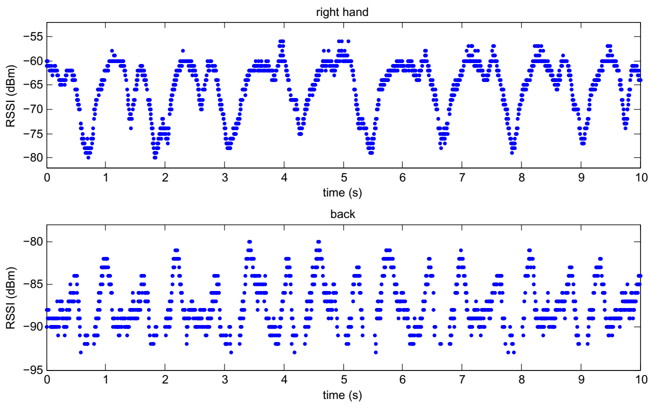


Fig. 2. RSSI measured on the right hand (top) and back (bottom) while the subject is walking

2.1 Experimental Setup: Shimmer2

In our setup, a BAN consists of eight Shimmer2 [24] nodes. Like the popular Telos [23] platforms, Shimmer2 integrates the Texas Instruments MSP430 MCU and the IEEE 802.15.4-compliant CC2420 transceiver [4]. The Shimmer2 platform also incorporates a Bluetooth radio, but we don't use it. Our Shimmer2 nodes are also equipped with a 2 GB MiniSD card, which was sufficient to store all traces that accumulated during one set of experiments.

The nodes were positioned on the subjects as shown in Fig. 1. An experiment consisted of one node (*sender*) continuously broadcasting IEEE 802.15.4 packets with a constant transmission frequency of 200 Hz – one 14-byte (MPDU size) packet every 5 ms. The other seven nodes (*receivers*) were passively listening for these packets (they did not send acknowledgments). The sender used a transmission power of -10 dBm[1], and it was always located in the right trouser pocket of the subject[2]. Our measurement software accesses the CC2420 radio directly, i.e., there is no MAC layer involved and the senders send packets immediately without clear channel assessment (CCA). Each of the seven receiver nodes keeps statistics of the number of correctly received packets and the associated Received Signal Strength Indicator (RSSI), which is measured using the first eight symbols following the start-of-frame delimiter (SFD) of the received packets [4]. The receiver nodes are placed on the left and right ankle, left trouser pocket, left and right hand, in the center of the chest, and in the center of the back (Fig. 1).

[1] The CC2420 supports a transmission power of 0 dBm, but previous work [13] has shown that -10 dBm often results in acceptable packet reception rates.

[2] In real life this node could be the user's cell phone, which is often carried around this location.

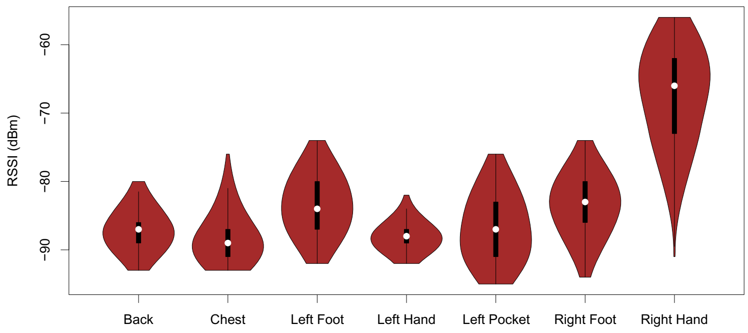


Fig. 3. Typical RSSI fluctuations during a 5-minute experiment

In each experiment the subjects walked continuously outdoors in a large park, an environment of negligible external RF interference, as we verified with the help of periodic noise-floor measurements on the nodes. The subjects were walking at the speed of approximately 1.2 steps/s, which is a typical walking speed. A single experiment lasted for 5 minutes (60,000 packets). Three different subjects performed 10 experiments each.

RSSI Fluctuations. While a subject was walking, the changes in the relative positions of the limbs manifested as periodic fluctuations in the RSSI. For example, the top graph in Fig. 2 shows a 10-second snapshot of the RSSI obtained in one experiment on a node that was positioned on the right hand of the subject (recall that the sender is always located in the right trouser pocket). The graph shows a period of about 1.2 s, which matches the step frequency of the subject. It also shows the frequent occurrence of a a plateau of about 1 s duration with RSSI values of −60 dBm followed by short trough with RSSI values as low as −80 dBm, corresponding to a significant RSSI range of approximately 20 dB. The node position, however, has an impact on the RSSI pattern: for example, the RSSI time series obtained in the same experiment on the back of the subject is much more noisy (Fig. 2 bottom).

Our goal is to exploit the RSSI fluctuations by scheduling packet transmissions such that they occur when the RSSI values are high, because then the chances are better that the packet are received correctly [25]. One precondition is that there is enough variance in the RSSI time series. To get an estimate of the magnitude of the RSSI fluctuations, we examined the RSSI inter-quartile ranges (IQR): for every 5-minute experiment we determined the distance between the 75th percentile and the 25th percentile of the RSSI readings per node position, essentially the range of the middle 50% of the data. Figure 3 summarizes the results in a violin plot (a combination of a boxplot and a kernel density plot), where the IQRs are shown as thin black boxes around the median (white dot). Thus the edges of the boxes represent the 25th and 75th percentiles. In this experiment the RSSI IQR varied between 2 and 9 dB. Note that RSSI values below the −94 dBm sensitivity threshold of the radio [4] are unavailable because

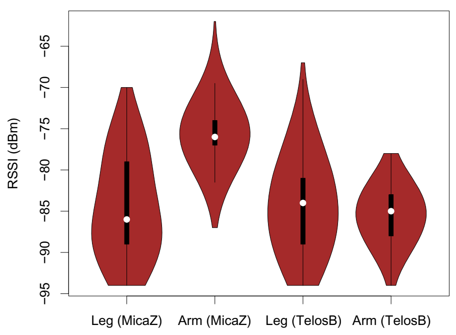


Fig. 4. Typical RSSI fluctuations on TelosB and MicaZ platforms

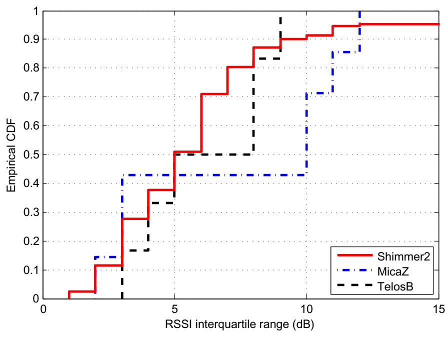


Fig. 5. CDF of the RSSI inter-quartile ranges

these packets are typically dropped. Figure 3 can be regarded as a representative example for the 10 experiments carried out by that specific subject, because in all experiments performed by a certain subject we found the corresponding median and IQR to be usually very similar.

The larger the RSSI IQR the more spread out are the RSSI values and thus the higher the potential for exploiting fluctuations by timely packet scheduling. Figure 5 shows a CDF of the RSSI IQR for all 30 experiments including the data of all three subjects. It can be seen that half of the links have an RSSI IQR of at least 5 dB and on 20% of the links the RSSI IQR is at least 8 dB.

2.2 Experimental Setup: MicaZ and TelosB Platforms

We repeated a set of similar experiments on the TelosB and MicaZ platform. Like Shimmer2, both platforms are also equipped with the CC2420 radio, but they have different antennas: while Shimmer2 has an SMD antenna, TelosB features inverted-F microstrip antenna and MicaZ features a half-wave dipole antenna. One of the goals was to investigate the differences arising due to different types of antennas. In these experiments we reduced the transmission rate to 20 packets/s and set the transmission power to -20 dBm. We used only one sender/receiver pair at a time. The sender was positioned on either the right foot or the right upper arm and the receiver was positioned on the chest. The receiver forwarded the received packets over the serial port to a laptop. The subject was walking at normal walking speed outdoors on a lawn.

In the following, we use MA to indicate the experiments with MicaZ nodes where the sender was placed on the upper arm and TA to indicate the same experiments where the nodes were TelosB motes. Similarly, we use ML to indicate the experiments with MicaZ nodes where the sender was placed on the leg and TL to indicate the same experiments where the nodes were TelosB motes. We also performed these experiments where the subject stood still, which we label as “MA Static” and so on.

RSSI Fluctuations. Fig. 4 shows violin-plots of some representative results (again in all experiments performed by a given subject the median and IQR per node position are usually very similar) and Fig. 5 shows the IQR CDFs for the two platforms. We found an IQR link margin of approx. 5 dB for the chest-arm pairs and of approx. 10 dB for chest-leg pairs (Table 1). Thus, aside from some differences due to different transmission power levels, the RSSI IQRs on the TelosB and MicaZ platforms are similar to those obtained on the Shimmer2 platform, which confirms that the effect is not platform-specific, but a general one.

Due to mobility, the standard deviations of RSSI fluctuations increased by approx. 1 dB when the sender was placed on the upper arm (Table 1). This fluctuation increased to 2.9 dB for MicaZ and 1.4 dB for TelosB motes when the sender was placed on the leg. However, mobility does not necessarily results in the decrease of the mean or median of the RSSI fluctuations. Contrary to the indoor measurements reported in [6], we found comparable attenuation (in ML and TL configurations) using printed (TelosB) and dipole (MicaZ) antennas, which can be explained by the absence of multi-path receptions outdoors.

Table 1. Summary of RSSI fluctuations

	RSSI			
Expt. Config.	Median Gain (dB)	IQR (dB)	Range (dB)	Std. Dev (dB)
MA	-51	4	31	2.81
MA Static	-55	2	27	1.79
TA	-65	5	17	3.68
TA Static	-62	5	21	2.87
ML	-65	12	26	6.19
ML Static	-62	4	21	3.33
TL	-64	9	37	5.56
TL Static	-67	7	20	4.11

2.3 Discussion

The RSSI fluctuations are influenced by several factors. The periodic changes in the relative positions of the limbs causes periodic differences in (1) relative node distance, which influences path-loss and fading, (2) shadowing and (3) relative node orientation. All these, in conjunction with the irregular antenna radiation pattern of the nodes, can result in different signal strength at the same distance. Our evaluation of the RSSI IQRs revealed that the magnitude of the RSSI fluctuation is usually position-dependent and often *significant* (several dB). In addition, the absolute RSSI values were often close to the sensitivity threshold of the radio, where even a small change in RSSI can result in a substantial difference in packet delivery performance [25]. However, in order to exploit any fluctuation, the RSSI pattern must also be *predictable*. Ideally, it should be periodic

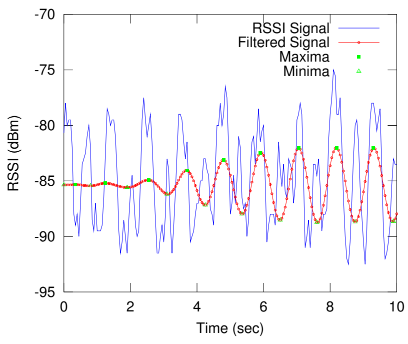


Fig. 6. Bandpass filtered RSSI signals – TelosB

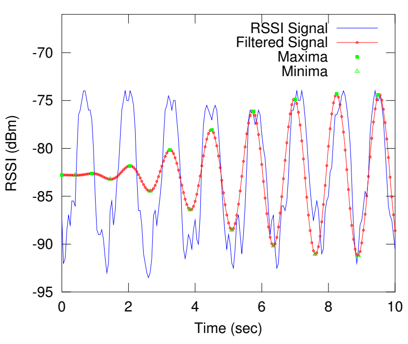


Fig. 7. Bandpass filtered RSSI signals – MicaZ

and maintain durable amplitudes as shown in the top graph in the top Fig. 2, because then the packets can be scheduled within the RSSI peak time windows and the packet losses can be reduced.

3 Opportune Transmission Windows

We use the term *opportune transmission window* (OTW) to describe a time interval that yields high RSSI values relative to the average RSSI of the link. Assuming that the subject performs regular movements, intuitively, we can predict an OTW by adding the current step period to the time of the previous OTW center. In this section we describe a method that derives both of these from RSSI time series obtained from a set of initial probe (control) packets.

3.1 RSSI-Based OTW Prediction

The main difficulty in using RSSI measurements to predict opportune transmission windows arises due to significant noise content in the RSSI measurements (Section 2.1). The second challenge arises due to the irregularities of human movements, which are usually never exactly periodic. Consequently, the simplistic approach of locating the peaks and extrapolating the inter-peak separation to predict OTW fails. However, we observed that in the Fourier domain the dominant peak in the power spectrum of RSSI time series corresponds to the speed of the subject.

RSSI-based OTW Prediction Algorithm. In order to find the OTW the sender transmits RSSI probe packets which are received at an appropriate node specified by the coordinator. The probe packets are transmitted at (sufficiently) low frequency, interspersed between data packets. The receiver returns the RSSI values of the probe packets in aggregated form to the coordinator, possibly piggybacked on the data packets. The coordinator maintains a moving window of

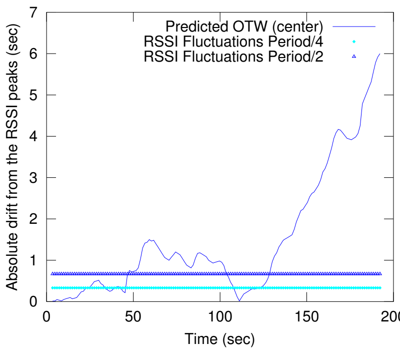


Fig. 8. Deviations of OTW predictions without dynamic correction

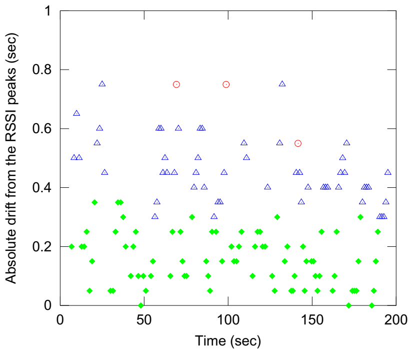


Fig. 9. Deviations of OTW predictions with periodic corrections: MicaZ sample

the RSSI time series. At the coordinator node, we apply fast Fourier transform (FFT) to the RSSI time series and find the dominant frequency. To determine the phase, we first apply a tight bandpass filter centered at the dominant frequency. Figures 6-7 show the filtered signal superimposed on samples of raw RSSI data. On the filtered signal, we then apply an extrema identification algorithm to determine the peaks. Since the RSSI sample has arbitrary phase at the two ends, we select the last but one peak as the basis for OTW predictions, to which we add integral multiples of the period (1/dominant frequency) for one set of nodes and odd half integral multiples of period for the other set of nodes, where the nodes on the left hand and right leg constitute one of the two sets, and symmetrically the nodes on the left hand and the right leg constitute the other. The first set is defined by the membership of the node that provides the RSSI samples. The rectification of the drifts in OTW predictions due to irregularities in the subject's movements can be done by re-running the algorithm either periodically or on-demand, for example, when significant deviation from the predictions are detected.

3.2 Evaluation

For evaluating the OTW prediction algorithm, we used the measurement data described earlier in Section 2.2. Our bandpass filter uses Butterworth filters. The filter pass frequencies were set to 0.1 Hz below and above the dominant frequency. The cutoff frequencies were 0 Hz and twice the central frequency. The passband ripple was set to 1 dB and the stopband attenuation was set to 30 dB.

Figure 8 shows the RSSI peak prediction drifts, or, the absolute difference between the peaks of the filtered RSSI signal and the center of predicted OTWs. For this figure, the expected times of RSSI peaks were obtained by adding multiples of the period obtained initially. The deviations are the differences of the nth RSSI peak prediction time and the actual nth peak observed in the filtered

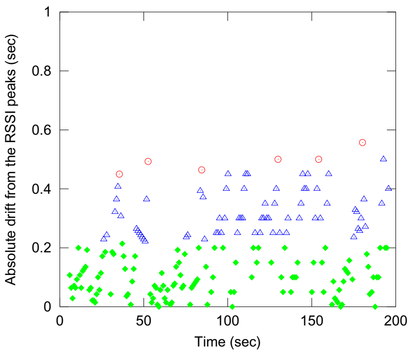


Fig. 10. Deviations of OTW predictions with periodic corrections: Shimmer2 sample

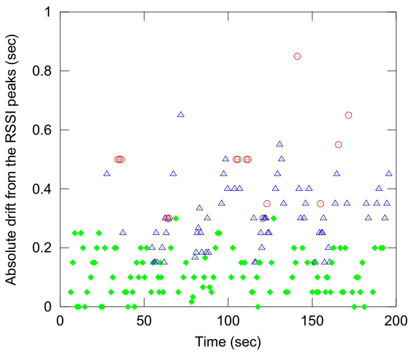


Fig. 11. Deviations of OTW predictions with periodic corrections: TelosB sample

data. Due to varying pace, stopping and other irregularities in walking, the drift generally grows with time.

The next three figures (Fig. 9–11) show the absolute deviations between the center of predicted OTWs and the nearest bandpass filtered RSSI peaks observed in the experiments, when the predictions were periodically adjusted. We sampled RSSI periodically every 12 s and collected RSSI samples at 20 Hz for 4.5 s. In other words, we used approx. one-third of the RSSI time series to validate this algorithm. The sampling time of 4.5 s was chosen to ensure the inclusion of at-least one pair of consecutive RSSI peaks. In figures 9–11, the drifts less than $0.25 * period$ are shown with diamonds, those less than $0.5 * period$ and greater than $0.25 * period$ with triangles and the larger drifts with circles. The means of the absolute deviations were 0.28 s for the MicaZ samples, 0.21 s for the TelosB samples and 0.18 s for the Shimmer2 samples. In all three cases, the non-central node was on one of the legs.

We observed that both the probe frequency and the probe duration can be significantly reduced without sacrificing accuracy of predictions. The optimization of these parameters is one of our future work. We note that if the coordinator and at-least one of the nodes have more than one radio transceiver, such as in the Shimmer2 motes, then our OTW prediction method can also be applied using secondary radios.

4 BANMAC

In this section we illustrate the use of the work presented in the previous two sections. We present BANMAC, a MAC protocol for body area networks which attempts to schedule transmissions during the time windows when the RSSI is expected to be larger than average. Our network model consists of a set of nodes connected in star topology to a coordinator node where the coordinator is significantly more powerful than the rest of the nodes.

Similar to IEEE 802.15.4, BANMAC alternates between centralized and distributed medium access modes. In the centralized mode, where the scheduling decisions are made by the coordinator, the channel access is collision-free and the MAC protocol supports features such as priority and guaranteed data rate. In the distributed scheduling mode of BANMAC, the nodes determine the time windows for opportunistic transmissions locally and contend for channel access during these windows. We describe the centralized scheduling algorithm of BANMAC in the following subsection (4.1). A detailed presentation of BANMAC will appear in a separate publication.

4.1 Centralized BANMAC Scheduling

Opportune Transmission Windows (OTW). Recall that we use the term *opportune transmission window* (OTW) to describe a time interval that yields high RSSI values relative to the average RSSI of the link, for example, the time interval of 3.5 s to 4.0 s in the top graph in Fig. 2. Assuming that the period and phase of the RSSI fluctuations are known (we describe an algorithm that derives this information from the RSSI time series in the previous section), let T be the period of RSSI fluctuations. Let $t = 0$ correspond to the start of the positive half-cycle (zero phase) for some node i as shown in Fig. 12. Then, the OTWs of node i span $[nT+(T/4-\Delta/2), nT+(T/4+\Delta/2))$ where n is an integer and Δ is the width of the OTWs. The left hand and right leg move in synchrony, and so do the other pair. Due to the synchronous and alternating motion of the pair of limbs, the time axis contains alternating OTWs, each OTW for the set of nodes on the two limbs that move together. These are shown as $\Delta(S_1)$ and $\Delta(S_2)$ in the figure, where node i belongs to the set S_1. Observe that $\Delta(S_1)$ and $\Delta(S_2)$ need not overlap. The gaps may be used for communication with nodes whose RSSI do not exhibit periodic fluctuations and for distributed medium access.

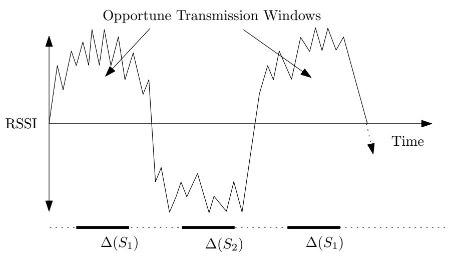


Fig. 12. Opportune transmission windows alternate for the two sets of nodes S_1 and S_2, explained in the text

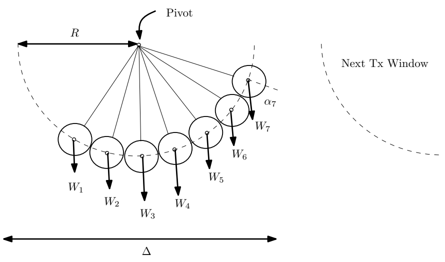


Fig. 13. Pendulum task model

Pendulum		Opportunistic Transmissions	
Name	Symbol	Name	Expression
Pivot	P	Target Tx time	$t_{\mathcal{T}}$
Weight of bob	W_i	Importance	w_i
Diameter of bob	e	Tx duration	e
Swing range	$2R$	Opportune Tx window	$\Delta - e$

Fig. 14. Mapping of opportunistic transmissions parameters to Gravitational Task Model

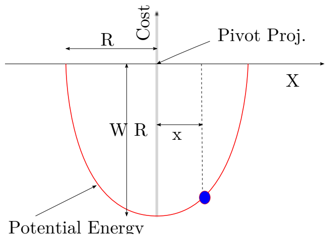


Fig. 15. Cost at displacement x from pivot (OTW center) in the Gravitational Task Model

Scheduling

We associate a cost function with the time of transmissions where the minima of the cost function coincide with the center of the OTWs. In the following, we present an algorithm that minimizes the cost of a set of transmissions. We adapt the Gravitational Task Model of Guerra and Fohler [10] to solve this problem. Their work is inspired from the physical system of a set of pendulums in equilibrium (Fig. 13), where the equilibrium is characterized by the minimum total potential energy of the set of pendulums.

Problem Formulation. Let us consider a set of n transmissions $\mathcal{T} = \{X_1, \ldots X_n\}$ to be scheduled during a given opportune transmission window, where X_i denotes the events of polling by the coordinator followed by the transmission of data packet to the coordinator. It may be that a number of data packets are sent to the coordinator following one poll packet. For the sake of simplicity of presentation, we treat the transmission of m data packets in succession as m single transmissions. Let $t_{\mathcal{T}}$ denote the center of the OTW and let Δ denote the width of the OTW. Let w_i denote the numerical measure of the importance of X_i. Finally, let e be the time needed to complete a single transmission event, X_i.

We map our problem formulation to a Gravitational Task Model instance as follows: we consider a set of n pendulums hanging from a pivot (Fig. 13). A bob in the pendulum system corresponds to one transmission. The diameter of the bobs (the same for all bobs) map to the transmission duration e. The weights of the pendulums correspond to the importance of transmissions. Furthermore, the swing range $2R$ maps to the width of OTW (minus a correction term), $\Delta - e$[3]. Table 14 presents a summary of the mappings.

The potential energy of a bob is U_i = weight * vertical displacement from the minimum $= -W_i\sqrt{R^2 - x_i^2}$ where x_i is the horizontal deviation of the bob from the central position, that is, the horizontal deviation from the projection

[3] We have simplified the model presented in [10] by redefining R as above.

of $t_{\mathcal{T}}$ (Fig. 15). The minimization of the potential energy can be expressed as the following nonlinear program (NLP):

$$\text{Minimize} \quad \sum_{i=1}^{n} -W_i\sqrt{R^2 - x_i^2}, \tag{1}$$

$$\text{subj. to} \quad x_{i+1} - x_i = e, \ i = 1, \ldots n-1, \tag{2}$$

$$\text{where} \quad |x_i| \leq R, \ i = 1, \ldots n-1. \tag{3}$$

In (2), the equality sign (rather than $\geq$) is chosen because all transmissions within any given window have the same target transmission time, and hence, the optimal solution can't have gaps. We solve this NLP in *linear time-complexity* by using the physical facts that in equilibrium, the bobs touch each other and the net torque on the system is zero. The interested reader is referred to [10] for more details. A transmission schedule is given by the projection of bobs on a horizontal line below the pendulums. However, the projections, if not corrected, overlap. In the following, we find expressions for the projections of the center of bobs sufficiently shifted such that the projections of the bobs do not overlap.

Minimum cost schedule. Let α_i be the angle that pendulum i makes with the vertical (Fig. 13). Then the displacement of the center of the bob along the X-axis is $x_i = R\sin\alpha_i$. Thus, we get

$$R\sin\alpha_{i+1} - R\sin\alpha_i = e, \ \forall i \tag{4}$$

$$\sum_{i=1}^{n} W_i R\sin\alpha_i = 0. \tag{5}$$

Eqn. 4 can be re-written as:

$$R\sin\alpha_i = R\sin\alpha_n - (n-i)e. \tag{6}$$

Substituting (6) in (5), we get the expression for the displacement of the last bob in equilibrium:

$$R\sin\alpha_n = \frac{\sum_{i=1}^{n-1}(n-i)eW_i}{\sum_{i=1}^{n} W_i} \tag{7}$$

The conditions $\sin\alpha_n > (1 - \frac{e}{2R})$ in (7) and $\sin\alpha_1 < -(1 - \frac{e}{2R})$ in (6) indicate the unschedulability of $\mathcal{T}$ within the opportune transmission window.

Examples. Let us consider a set of four pendulums, which models a set of four transmissions, say $\mathcal{T}$, to be scheduled in a given OTW. From (7), we get,

$$x_4 = R\sin\alpha_4 = \frac{e(3W_1 + 2W_2 + W_3)}{W_1 + W_2 + W_3 + W_4}. \tag{8}$$

Consider the case of equal weight bobs, i.e., $W_i = W \ \forall i$. Then, the displacement of the fourth pendulum from the center is $R\sin\alpha_4 = 1.5e$. We use (6) to get the displacements of other pendulums. The displacement of the third pendulum is $R\sin\alpha_3 = 0.5e$, that of the second pendulum is $R\sin\alpha_2 = -0.5e$ and that of the first pendulum is $R\sin\alpha_1 = -1.5e$. Thus the first transmission is scheduled at $t_{\mathcal{T}} - 2e$, the second transmission is scheduled at $t_{\mathcal{T}} - e$ and so on. Now suppose $W_1 >> W_i,\ i = 2, 3, 4$. Then, from (8), $R\sin\alpha_4 \approxeq 3e$, which corresponds to the case of pendulum 1 being almost at the center as expected.

The work presented in [10] (and Equation 4) minimizes the cost function for a given ordering of transmissions. From symmetry, the ordering that maximizes the number of schedulable transmissions corresponds to the ordering where the "center of mass" is located in the middle. However, the possibility of performing such orderings is limited by the start time constraints and transmission order dependencies.

5 Related Work

Several studies have focused on the effects of human body shadowing on RF communication. Kara et al. describe an experimental study that evaluates attenuation due to people crossing a 2.4 GHz band link [15]. They show that the human body can result in attenuation of up to 20 dB. Shadowing effects caused by a human body have been studied in [8] for 802.11 radios. There are also several studies that focus on human body shadowing at other frequencies. For example, the 900 MHz and 60 GHz bands are the focus of [18] and the 10 GHz band is the focus of [9].

Due to its focus on low-power communication, the work of Miluzzo et al. [17] is closely related to ours. The authors performed an experimental study in which they investigate person-to-person communication with IEEE 802.15.4 radios. Their results indicate that the position of the radio on the human body as well as the attenuation introduced by the human body has a significant effect on the performance of the communication. The experimental part of our work can be regarded complementary as we have focused on intra-BAN communication only. Some of the work presented in IEEE 802.15.6 WG proceedings is also related to ours. Davenport et al. present a study of link characterization of medical BAN indoors [6]. In [2], Cai et al. derive a two state channel model based on empirical RSSI measurements in BANs, which also match our experimental results. A MAC protocol for BANs is proposed in [30], where throughput maximization is the objective. To the best of our knowledge, our work is the first to propose the idea of opportunistic transmission scheduling exploiting RSSI fluctuations for better reliability.

Like BANs, energy efficiency is one of the main concerns in Wireless Sensor Networks (WSN). A number of energy efficient MAC protocols have been proposed for WSNs. Putting nodes to sleep is the primary mechanism for energy saving in S-MAC [28] and T-MAC [5]. Their main difference is the use of fixed versus variable sleep cycles (see [11] for a comparative study) . Since all receivers listen during wake-up periods when the energy spent while not receiving transmissions is wasted, low power listening (LPL) at the receivers improves energy efficiency. Berkeley-MAC [19], WiseMAC [7] and X-MAC [1] use this paradigm.

Large latency is a critical problem with this approach. X-MAC improves energy efficiency of B-MAC and WiseMAC by strobing the long preambles and inserting receiver ID in the strobes. An early CTS in X-MAC alleviates the latency problem. CMAC [16] also tries to improve the LPL latency. However, the protocol has the overhead of synchronizing time slots. Scheduled channel polling (SCP) [29] eliminates the need for long preambles in LPL by synchronized polling. At the time of polling, a sender wakes up the receiver and after performing a short carrier sense, it transmits the packet. SCP has been found to decrease the limiting duty cycle LPL from 1-2% to 0.1%. TDMA based MAC protocols can offer bounded delays and collision-free transmissions. TRAMA [20] is a collision-free TDMA MAC protocol for WSN. ZMAC [22] is a hybrid MAC protocol, that operates in CSMA mode under light load conditions and in TDMA mode under heavy load conditions.

Targeted scheduling of tasks have been studied before by the real-time systems research community. Time Value Functions (TVF) or, Time Utility Functions (TUF) express the value of the completion of a task as a function of time. Jensen et al. proposed the use of TVFs for real-time process scheduling [14]. Chen and Muhlethaler present a heuristic to schedule tasks that attempts to maximize the sum of TVF for each task [3]. Wang and Ravindran improve the heuristic of Chen and Muhlethaler from $O(n^3)$ time complexity to $O(n^2)$ time complexity [27]. See [21] for a survey.

6 Conclusion

In an experimental study conducted with three mote platforms, we showed that regular human movements often manifest in significant periodic RSSI fluctuations. We presented and evaluated an algorithm that predicts opportune transmission windows and deals with the issue of irregularities of human movements and noisy RSSI signals. We also presented a sketch of a MAC protocol for BANs, called BANMAC, that takes advantage of these fluctuations by opportunistically scheduling transmissions when the RSSI is likely to be higher than the average for better reception reliability. In an ongoing work, we are integrating BANMAC to the IEEE 802.15.4 protocol stack. We plan to extend our work to incorporate dynamic power control and channel migration in the near future.

Acknowledgments. This research was partially supported by the European Commission under contract number FP7-2007-IST-2-224053 (CONET) and by the Science and Technology Foundation of Portugal (FCT). We thank Claro Noda and Jasper Büsch for helping as the subjects of some of the experiments.

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: SenSys 2006: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, pp. 307–320. ACM, New York (2006)

2. Cai, J., Cheng, S., Huang, C.: MAC channel model for WBAN. Tech. Rep. 15-09-0562-00-0006, IEEE P802.15 (July 2009)
3. Chen, K., Muhlethaler, P.: A scheduling algorithm for tasks described by time value function. Real-Time Syst. 10(3), 293–312 (1996)
4. Chipcon Corporation: CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver (April 2002), `http://www.ti.com/lit/gpn/cc2420`
5. 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)
6. Davenport, D., Ross, F., Deb, B.: Wireless propagation and coexistence of medical body sensor networks for ambulatory patient monitoring. In: International Workshop on Wearable and Implantable Body Sensor Networks, pp. 109–113. IEEE Computer Society, Los Alamitos (2009)
7. El-Hoiydi, A., Decotignie, J.D.: Low power downlink MAC protocols for infrastructure wireless sensor networks. Mob. Netw. Appl. 10(5), 675–690 (2005)
8. Gaertner, G., ONuallain, E., Butterly, A., Singh, K., Cahill, V.: 802.11 link quality and its prediction – an experimental study. In: Niemegeers, I., de Groot, S.H. (eds.) PWC 2004. LNCS, vol. 3260, pp. 609–611. Springer, Heidelberg (2004)
9. Ghaddar, M., Talbi, L., Denidni, T.A.: Human body modelling for prediction of effect of people on indoor propagation channel. Electronics Letters 40(25), 1592–1594 (2004)
10. Guerra, R., Fohler, G.: A gravitational task model with arbitrary anchor points for target sensitive real-time applications. Real-Time Syst. 43(1), 93–115 (2009)
11. Halkes, G.P., Van Dam, T., Langendoen, K.G.: Comparing energy-saving MAC protocols for wireless sensor networks. Mob. Netw. Appl. 10(5), 783–791 (2005)
12. Hanson, M.A., Powell Jr., H.C., Barth, A.T., Ringgenberg, K., Calhoun, B.H., Aylor, J.H., Lach, J.: Body area sensor networks: Challenges and opportunities. Computer 42, 58–65 (2009)
13. Hauer, J.H., 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)
14. Jensen, E.D., Locke, C.D., Tokuda, H.: A time-driven scheduling model for real-time operating systems. In: RTSS 1985: Proc. of the 6th IEEE Real-Time Systems Symposium, pp. 112–122. IEEE Press, Los Alamitos (1985)
15. Kara, A., Bertoni, H.: Blockage/shadowing and polarization measurements at 2.45 ghz for interference evaluation between Bluetooth and IEEE 802.11 WLAN. In: IEEE Antennas and Propagation Society International Symposium, vol. 3, pp. 376–379 (2001)
16. Liu, S., Fan, K.W., Sinha, P.: CMAC: an energy-efficient MAC layer protocol using convergent packet forwarding for wireless sensor networks. ACM Trans. Sen. Netw. 5(4), 1–34 (2009)
17. Miluzzo, E., Zheng, X., Fodor, K., Campbell, A.T.: Radio characterization of 802.15.4 and its impact on the design of mobile sensor networks. In: Verdone, R. (ed.) EWSN 2008. LNCS, vol. 4913, pp. 171–188. Springer, Heidelberg (2008)
18. Obayashi, S., Zander, J.: A body-shadowing model for indoor radio communication environments. IEEE Transactions on Antennas and Propagation 46(6), 920–927 (1998)
19. 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)

20. Rajendran, V., Obraczka, K., Garcia-Luna-Aceves, J.J.: Energy-efficient collision-free medium access control for wireless sensor networks. In: SenSys 2003: Proc. of the 1st International Conference on Embedded Networked Sensor Systems, pp. 181–192. ACM, New York (2003)
21. Ravindran, B., Jensen, E.D., Li, P.: On recent advances in time/utility function real-time scheduling and resource management. In: IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 55–60. IEEE Computer Society, Los Alamitos (2005)
22. Rhee, I., Warrier, A., Aia, M., Min, J.: Z-MAC: a hybrid MAC for wireless sensor networks. In: SenSys 2005: Proc. of the 3rd International Conference on Embedded Networked Sensor Systems, pp. 90–101. ACM Press, New York (2005)
23. Sentilla Corporation: Tmote sky datasheet, `http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf`
24. Shimmer Research: Shimmer2 capabilities overview, `http://www.shimmer-research.com/wpcontent/uploads/2010/08/Shimmer-2R-Capabilities-Overview.pdf`
25. Srinivasan, K., Levis, P.: RSSI is under appreciated. In: Proceedings of the Third Workshop on Embedded Networked Sensors, EmNets 2006 (2006)
26. U.S. Health Resources and Services Administrations: What is Behind HRSA's Projected Supply, Demand, and Shortage of Registered Nurses? (September 2004), `http://bhpr.hrsa.gov/healthworkforce/reports/behindrnprojections/index.shtm`
27. Wang, J., Ravindran, B.: Time-utility function-driven switched ethernet: Packet scheduling algorithm, implementation, and feasibility analysis. IEEE Transactions on Parallel and Distributed Systems 15, 119–133 (2004)
28. Ye, W., Heidemann, J., Estrin, D.: Medium access control with coordinated adaptive sleeping for wireless sensor networks. IEEE/ACM Trans. Netw. 12(3), 493–506 (2004)
29. Ye, W., Silva, F., Heidemann, J.: Ultra-low duty cycle MAC with scheduled channel polling. In: SenSys 2006: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, pp. 321–334. ACM, New York (2006)
30. Yoo, S.M., Chen, C.J., Chou, P.H.: Low-complexity, high-throughput multiple-access wireless protocol for body sensor networks. In: International Workshop on Wearable and Implantable Body Sensor Networks, pp. 109–113. IEEE Computer Society, Los Alamitos (2009)

Clock Synchronization with Deterministic Accuracy Guarantee

Ryo Sugihara and Rajesh K. Gupta

Computer Science and Engineering Department,
University of California, San Diego, California, USA
{ryo,rgupta}@ucsd.edu

Abstract. Accuracy is one of the most important performance metrics in clock synchronization. While state-of-the-art synchronization protocols achieve μsec-order average accuracy, they usually do not focus on the worst case accuracy and do not have any deterministic guarantees. This lack of accuracy guarantee makes it hard for sensor networks to be incorporated into larger systems that require more reliability than e.g., typical environmental monitoring applications do. In this paper, we present a clock synchronization algorithm with deterministic accuracy guarantee. A key observation is that the variability of oscillation frequency is much smaller in a single crystal than between different crystals. Our algorithm leverages this to achieve much tighter accuracy guarantee compared to the interval-based synchronization methods mostly proposed in the literature of distributed systems. We designed an algorithm to solve a geometric problem involving tangents to convex polygons, and implemented that in TinyOS. Experimental results show the deterministic error bound less than 9.2 clock ticks (280 μsec) on average at the first hop, which is close to the simulation results. Further, by a combination with previously proposed synchronization algorithms, it achieves the estimation error of 1.54 ticks at 10 hop distance, which is more than 40% better than FTSP, while giving deterministic error bounds.

1 Introduction

Clock synchronization is a fundamental service that is required in many applications in sensor networks as well as in distributed systems in general. It is of more importance when sensor networks extend beyond research-oriented systems and get incorporated into industrial systems often referred to as cyber-physical systems (CPS). These systems are often mission-critical, and thus providing a performance guarantee is as important as yielding a good average performance. In the context of clock synchronization, performance guarantee corresponds to guarantees on accuracy.

Accuracy guarantee in clock synchronization, especially deterministic one, is important in sensor networks in several situations including sensor fusion, coordinated actions, and hard-realtime applications, as discussed in [7]. We add security applications to the list. For example in secure localization (e.g., [8]), the accuracy of clock synchronization directly affects the reliability of the location estimation and thus the security level that the application can provide.

Recent research on clock synchronization in wireless sensor networks has been mostly focused on improving the average case accuracy rather than the worst case. Although state-of-the-art techniques enable impressive average performance of few microseconds error [14,15,21], they either do not have any guarantees or only have probabilistic guarantees on the worst case accuracy.

In this paper, we propose a novel clock synchronization algorithm that gives a deterministic accuracy guarantee characterized by upper and lower limits on the current time. Although the basic idea is analogous to classical interval-based clock synchronization [6,7,9,13,16,18,20], there are several key differences that enable our algorithm to achieve more correct and tighter bounds. One of the differences is the bounded drift fluctuation, which is given in the data sheet of crystal oscillators and we also experimentally verified by ourselves. The algorithm does not require or construct a fixed network topology and works completely in a distributed manner. It is also efficient by packing the information for multiple receivers into a single packet thus leveraging the broadcasting nature of wireless communication.

Our contribution is a novel clock synchronization algorithm that

– Gives deterministic accuracy guarantee without simplifying assumptions,
– Is fully distributed and does not require any a priori knowledge on the network topology or the topology being stationary, and
– Achieves good clock estimation when combined with other algorithms.

We implement the algorithm in TinyOS for testbed experiments and compare the results with simulation and FTSP.

The rest of the paper is organized as follows. In Section 2, we present the system model as well as the experimental results on temperature vs. clock frequency. We present main ideas of the synchronization algorithm in Section 3 and the details in Section 4. In Section 5, we discuss some of the issues that arise in implementing the algorithm in TinyOS. Section 6 presents the evaluation results from both simulation and testbed experiments. We overview related work in Section 7 and Section 8 concludes the paper.

2 System Model

We first define the clock model and describe the assumptions. Then we validate these assumptions through preliminary experiments on the frequency vs. temperature characteristics of crystals.

2.1 Clocks

We refer to the clock reading at node v as *localtime* at v and denote as s^v. There are one or more nodes that have access to accurate time e.g., through GPS. These nodes are called *roots* and their time is called *globaltime* t, which is also called "wall-clock time" or "physical time" in the literature. All other nodes are just referred to as *nodes*. Node v's clock has a *clock drift* $h_v(s^v)$, which is defined as the amount of increase in globaltime when the localtime is increased by unit amount. For each node, we define *clock function* f_v to give corresponding globaltime for each localtime as follows:

$$f_v(s^v) = \int_0^{s^v} h_v(\tau) d\tau + \delta_v,$$

where δ_v is a constant called *clock offset*. As a notational convention, we assume globaltime t_i corresponds to localtime s_i^v; i.e., $t_i = f_v(s_i^v)$.

The drift consists of *drift offset* $\overline{h_v}$ and *drift fluctuation* $\alpha_v(s^v)$, where the former is a constant and the latter is a time-varying function:

$$h_v(s^v) = \overline{h_v} + \alpha_v(s^v).$$

While these are not known in advance, we assume they satisfy the following two properties:

- Bounded drift offset: $1 - \eta \leq \overline{h_v} \leq 1 + \eta$,
- Bounded drift fluctuation: $|\alpha_v(s^v)| \leq \xi$,

where η and ξ are given constants. The values of η and ξ depend on the type of crystal oscillator and usually specified in or can be calculated from the data sheet. In normal temperature range, ξ is much smaller than η. Note that we do not assume any statistical properties for $\alpha_v(s^v)$ except that it is bounded by ξ. Also note that η and ξ are different from "drift bound" (usually denoted by ρ) that often appears in the literature. Since $1 - \rho \leq h_v(s^v) \leq 1 + \rho$ by definition, we can consider that $\rho = \eta + \xi$. This is one of the key differences between the proposed algorithm and so-called interval-based methods, which we will discuss later in Section 3.4.

The objective of a synchronization algorithm is for each node to obtain a mapping from its localtime to the globaltime. We specifically focus on giving a deterministic guarantee on accuracy: for any localtime, each node must be able to tell the interval that the globaltime is contained within.

2.2 Crystal Oscillator

The assumptions of bounded drift offset and bounded drift fluctuation are not common and also very important for our algorithm. For these reasons, we have measured these values in the actual nodes to assess how reasonable these assumptions are. Since temperature is the primary cause that affects the clock frequency [22], we measure the frequency under different temperatures.

Crossbow Telos (Rev. B) nodes use Citizen CMR200T for 32.768kHz crystal oscillator [3]. This is a tuning fork crystal unit [1] and is known to have a quadratic relation between frequency and temperature. The relation is expressed by $f = f_0(1 + \beta(T - T_0)^2)$, where f_0 is the nominal frequency, T_0 is the reference

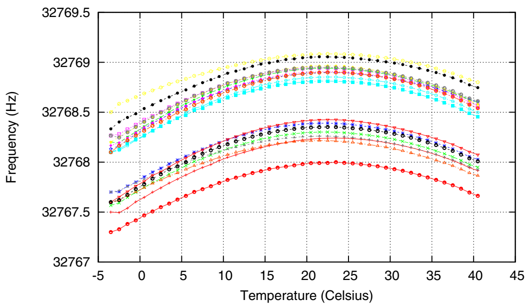


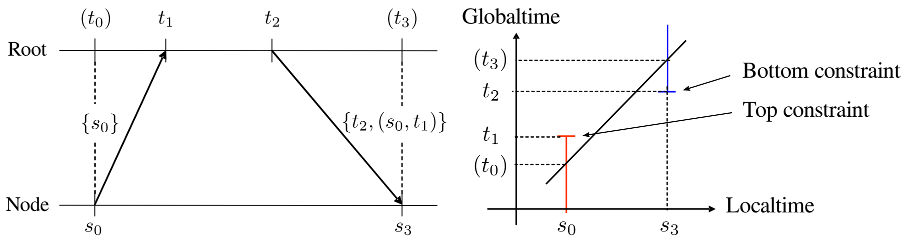
Fig. 1. Temperature vs. frequency of 32.768 kHz crystal oscillator: For 19 nodes

temperature, and β is a constant called "temperature coefficient." In practice the actual frequency f_0' at T_0 is different from f_0 by a small amount and the bound on frequency tolerance $(f_0' - f_0)/f_0$ is usually specified in the data sheet. For CMR200T, $f_0 = 32768$ Hz, $\beta = -0.034 \pm 0.006$ ppm/℃^{2} (ppm: parts-per-million), and the frequency tolerance is ± 20 ppm [1]. The characteristic differs by types and also by parameters of the cut.

Figure 1 shows the relation of temperature and frequency measured at 19 Telos B nodes. Clock frequency is measured by counting the number of ticks precisely in 10 seconds, which is obtained from PPS (Pulse-per-Second) output of a GPS module (Garmin 18x LVC). This GPS module guarantees that the PPS signal is aligned to the start of each GPS second within 1 μsec [2]. We gradually change the environment temperature and associate the measured clock frequency with the temperature measured simultaneously by the onboard sensor.

All nodes exhibit curves peaked between 20 to 25 ℃ and their shapes are close to the one specified in the data sheet (not shown). As for the offset, the peaks are above the nominal frequency (32768 Hz) in 18 out of 19 nodes. This is likely to be due to the mismatch between ideal and actual load capacitance. Specifically, CMR200T requires load capacitance of 12.5 pF, whereas the MPU (MSP430F1611) has internal 12 pF fixed capacitance per pin [5] and they are added serially for two pins, resulting in 6 pF capacitance in total. Smaller load capacitance leads to higher oscillation frequency [4] and apparently, it is in accordance with the results. Unfortunately, since there are other sources of parasitic capacitance e.g., from PCB traces, we do not have a guarantee as strong as that for the frequency tolerance in the crystal's datasheet. However, based on the observation, the curve still satisfies the frequency tolerance specification and we assume the frequency is only shifted by unknown small constant amount.

In summary, as we expected, we have observed larger variability among different crystals than in a single crystal at different temperatures. To accommodate the shift of peak frequency, we set the nominal frequency $f_0 = 32768.5$ [Hz]. In the later experiments, we assume the temperature range is 10 to 35 ℃. Then we can guarantee the frequency range for a single crystal is within 10 ppm (0.33 Hz in 32 kHz crystal) from both the specification and the measurement results. Since the peak frequency is within 20 ppm from the nominal frequency, we set



(a) Message exchange: t_0, t_3 are not known. (b) Constraints on clock function

Fig. 2. Basic idea for synchronization between a root and a node

the drift offset bound η to 25 ppm and the fluctuation bound ξ to 5 ppm to cover the whole range. If the temperature range is broader or unknown, these parameters must be chosen accordingly and conservatively to assure the correctness of the accuracy guarantee.

3 Synchronization with Accuracy Guarantee

In this section we describe the overall idea of our synchronization algorithm. For clarity we first assume constant drift (i.e., no drift fluctuation: $\xi = 0$) and explain the algorithm for synchronization between a root and a node. Then we extend it for synchronization between two nodes to enable network-wide synchronization, and also describe how we can take into account the drift fluctuation. Finally we discuss the differences between our algorithm and interval-based methods.

3.1 Main Idea

Our synchronization algorithm is based on a simple causality principle that a message is received only after it is sent. From sender and receiver timestamps, each node obtains a set of constraints that its clock function must satisfy.

Figure 2(a) shows the message exchange between a root and a node. The node sends a message at localtime s_0 and the root receives it at globaltime t_1. Then the root sends back a message at t_2 with the information on previously received message (s_0, t_1) as well as with the timestamp t_2. Upon receiving the message, the node can do the following calculations. For the first message, since the receiving time is later than the sending time, $t_0 \leq t_1$, though we cannot know t_0. Using $f(s_0) = t_0$, where f is the clock function for the node, we have $f(s_0) \leq t_1$. Similarly for the second message, we obtain $f(s_3) \geq t_2$.

The constraints that give upper or lower bounds for a clock function are called *top constraints* and *bottom constraints*, respectively (Fig. 2(b)). A constraint C_i is expressed by $(s_i, l_i, type_i)$, where s_i is localtime, l_i is called the *value* of C_i, and $type_i$ is either "top" or "bottom."

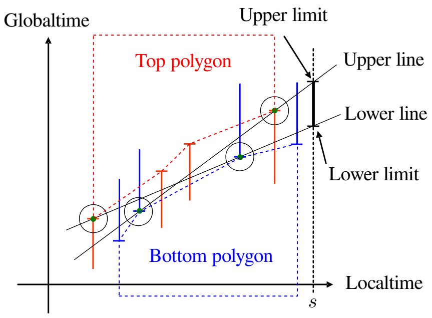


Fig. 3. Upper/lower limit given by upper-/lowermost line satisfying all constraints

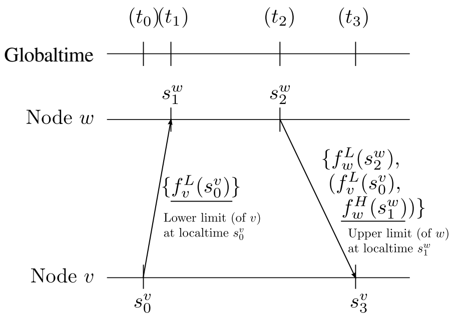

$(t_0)(t_1)$
(t_2)
(t_3)
Globaltime
Node w
Node v
s_1^w
s_2^w
s_0^v
s_3^v
$\{f_v^L(s_0^v)\}$
Lower limit (of v) at localtime s_0^v
$\{f_w^L(s_2^w), (f_v^L(s_0^v), f_w^H(s_1^w))\}$
Upper limit (of w) at localtime s_1^w

Fig. 4. Node-node message exchange: $t_0, ..., t_3$ are not known

After several messages are exchanged, each node has a set of top constraints and bottom constraints (Fig. 3). When we ignore the drift fluctuation (this is relaxed later in the section), f is a linear function that satisfies all the constraints. Of all such linear functions, we can determine the ones that give the maximum and minimum globaltime at given localtime s. We call these *upper line* and *lower line*, which are collectively called *limiting lines*. The maximum and minimum globaltime are called *upper limit* and *lower limit* at localtime s, respectively. Further, we call a constraint a *support* or a *support constraint* when it determines the upper or lower line.

The problem of finding limiting lines can be viewed in a more geometric way by considering two polygonal objects for each of the sets of top and bottom constraints. The clock function is a long bar and the upper and lower limits correspond to the range of motion when it is inserted between two polygons.

3.2 Network-Wide Synchronization

The algorithm for synchronization between a root and a node can be extended for network-wide synchronization. We describe the case for two nodes where neither of them is a root. We use f_v^H and f_v^L to denote the upper and lower limits at node v. For any localtime s^v, they satisfy $f_v^L(s^v) \le f_v(s^v) \le f_v^H(s^v)$.

Figure 4 shows the message exchange for synchronization between two nodes. Suppose node v initiates the message exchange. Different from the root vs. node case, at localtime s_0^v, v sends the lower limit $f_v^L(s_0^v)$ instead of s_0^v itself, which node v remembers for later use. Node w records the localtime s_1^w when it receives the message, but it remembers the upper limit $f_w^H(s_1^w)$ instead. Then in the response message, node w puts the pair $(f_v^L(s_0^v), f_w^H(s_1^w))$ as well as the lower limit $f_w^L(s_2^w)$, just as node v did. This pair of (lower limit at sent time, upper limit at received time) is called *SyncInfo* for the original sender.

After this message exchange, node v obtains the following two constraints:

$$\text{top: } f_v(s_0^v) = t_0 < t_1 = f_w(s_1^w) \le f_w^H(s_1^w) \tag{1}$$

$$\text{bottom: } f_v(s_3^v) = t_3 > t_2 = f_w(s_2^w) \ge f_w^L(s_2^w) \tag{2}$$

Note that $t_0, ..., t_3, f_w(s_1^w), f_w(s_2^w)$ are all unknown.

As a side-effect, node w also obtains a bottom constraint $f_w(s_1^w) > f_v^L(s_0^v)$ from the first message. This is a preferable property especially for wireless environment where every communication is essentially a broadcast. For network-wide synchronization, multiple receivers within the communication range of a sender can obtain a bottom constraint. We can also embed multiple SyncInfo in one message so that multiple receivers can obtain top constraints simultaneously.

3.3 Compensation for Drift Fluctuation

So far we have assumed that clock drift is constant. However, in practice, clock drift fluctuates over time with external causes, mostly due to temperature changes. Here we extend the algorithm for the case with drift fluctuations.

The idea is to compensate each of the constraints for the effect of drift fluctuation after the constraint is obtained. For constraint $C_i = (s_i, l_i, type_i)$, we define *compensated constraint* $\tilde{C}_i(s) = (s_i, \tilde{l}_i(s), type_i)$ at localtime s, where compensated value $\tilde{l}_i(s)$ is defined as follows:

$$\tilde{l}_i(s) = \begin{cases} l_i + \xi(s - s_i) & \text{if } type_i = \text{“top”} \\ l_i - \xi(s - s_i) & \text{if } type_i = \text{“bottom”} \end{cases}$$

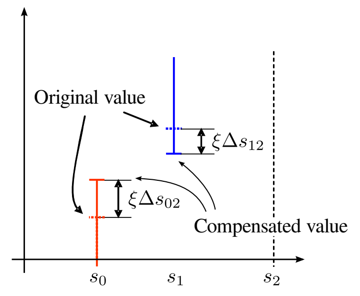
Then we have the following lemma:

Lemma 1. *Given $f(s)$ that satisfies all the constraints and $f(s_1) = t_1$, linear function $g(s) = \overline{h}s + \delta$ with $g(s_1) = t_1$ satisfies all the compensated constraints, where $\overline{h}, \delta$ is the drift offset and clock offset, respectively.*

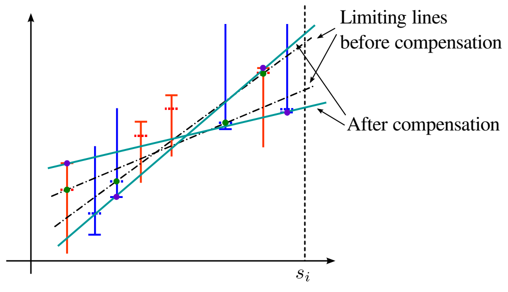
Proof is omitted from this version. Then we have the following theorem:

Theorem 1. *$\forall s. g^L(s) \le f^L(s) \le f^H(s) \le g^H(s)$, where $g^L(s), g^H(s)$ are the lower and upper limits at localtime s calculated for linear clock function $g(s)$ with the slope in $[1-\eta, 1+\eta]$ that satisfies all the compensated constraints.*

Figure 5 explains compensated constraints. As Fig. 5(a) shows, at localtime s_2, the value of the top constraint at s_0 is increased by $\xi \Delta s_{02}$, and that of the bottom constraint at s_1 is decreased by $\xi \Delta s_{12}$. Then, as shown in Fig. 5(b), we change the value for all constraints in the same way and the limiting lines are determined for these compensated constraints. Since the compensated constraints are “looser” than the original ones, according to Theorem 1, the new interval of upper and lower limits contains the one calculated without compensation. In this way, after compensation, we only need to solve the same problem of finding limiting lines.



(a) Compensation at localtime s_2



(b) Limiting lines at localtime s_i

Fig. 5. Compensated constraints

3.4 Comparison with Interval-Based Methods

Our algorithm shares the idea with the clock synchronization algorithms that provide an interval that the globaltime is contained. These algorithms, often called "interval-based algorithms," are proposed mostly in the literature of distributed systems before sensor networks emerged, but there are some papers in the contexts of mobile ad-hoc networks [18] and sensor networks [7] as well. Here we compare the proposed algorithm with these works.

While it is often claimed that an algorithm guarantees the accuracy, the correctness may not hold in the context of sensor networks. For example, some of the interval-based algorithms just ignore transmission delay (e.g., [7,16]). This is not an appropriate assumption when the transmission delay is dominant in the resulting accuracy. More often clock drift is assumed to be constant [6,9,23]. This is not appropriate as well for sensor networks that are often deployed outside and exposed to drastic temperature changes. Our algorithm does not make these assumptions and only uses the bounded drift offset and bounded drift fluctuation assumptions, both of which are guaranteed to be correct and derived from the data sheet of crystal oscillator.

Meanwhile, the common underlying idea for interval-based synchronization methods is to bound the actual length of any time interval by using most pessimistic value of clock drift. For example, when clock drift bound is ρ, we can guarantee that the actual length of localtime interval Δs is bounded by $[(1-\rho)\Delta s, (1+\rho)\Delta s]$. In our settings, we can emulate this by setting drift offset bound $\eta = 0$ and drift fluctuation bound $\xi = \rho$. This is because their assumption is precisely equivalent to assuming that each crystal can fluctuate in the whole range of $[1-\rho, 1+\rho]$. In the experiments, we will use this to compare our algorithm with the interval-based methods.

4 Algorithm Details

In this section we describe the algorithm in more detail. RECEIVE and SEND are the procedures called upon message reception and transmission, respectively.

Upon receiving a new message, constraints are added using AddConstraints and Convexify, and limiting lines are calculated using UpdateUpperLine/UpdateLowerLine. As we see later, all these procedures can be done in the time logarithmic to the number of constraints stored in a node.

4.1 Communications

A root broadcasts a message periodically. The period is to be determined based on application requirements and energy availability. In the experiments we use uniformly random period in $18 - 22$ seconds.

Upon receiving a new message, a node updates upper and lower lines. This is necessary because, due to the compensation of drift fluctuation, the limiting lines will change even without any changes in the sets of constraints.

After that, the node adds a bottom constraint based on the received time and the lower limit that the message has. The same information is saved as SyncInfo, which will be sent back to the sender. Then the node scans through all SyncInfo that the message carries. If there is one for this node, it adds a top constraint based on that.

Shortly after receiving, a node sends a message when one of the new constraints has become a support. We use this "pulse-like" propagation pattern based on the analysis in [14]. Upon sending a message, the node first updates the lower line to calculate the current lower limit.

4.2 Computations

Add Constraint. After adding a new constraint, we call Convexify to eliminate any non-convex constraints, since they will never become supports, as seen from Fig. 3. Then, if the new constraint violates the current limiting line, we update the limiting line.

Convexify. Convexify is the procedure for making the set of constraints form a convex polygon by eliminating non-convex constraints. Finding non-convex constraints is easily done by calculating the turning direction of two vectors (typically by using the external product) made by three points. This is similar to what Graham scan [10] does for calculating the convex hull of points. However, since in our case the points are already sorted and we know the set is convex without the new point, we can use binary search to find a convex point instead of linearly scanning the points. This makes the running time $O(\log n)$, when n is the number of top/bottom constraints stored in the node.

One thing to note is that, in case of top constraints, a newly added constraint may not be the latest constraint in terms of localtime. Therefore, we may need to run the above binary search for both directions from the inserted point, though it does not affect the order of the computation time.

```
procedure Receive($f_w^L, \{S\}$)    ▷ at localtime $s_{recv}^v$; $\{S\}$: Set of SyncInfo in received message
    UpdateUpperLine
    UpdateLowerLine
    AddConstraint($s_{recv}^v, f_w^L, bottom$)
    $f_v^H \leftarrow$ CalcUpperLimit
    if $f_v^H \neq \infty$ then SaveSyncInfo($src = w, f_w^L, f_v^H$)
    for $S_i \in \{S\}$ do
        if $S_i$ is for this node then
            $s_{send}^v \leftarrow$ RetrieveSendTime($S_i$)
            AddConstraint($s_{send}^v, value(S_i), top$)
    if new constraint became a support then Send

procedure Send                                        ▷ at localtime $s_{send}^v$
    UpdateLowerLine
    $f_v^L \leftarrow$ CalcLowerLimit($s_{send}^v$)
    StoreSendTime($s_{send}^v, f_v^L$)
    SendMessage($f_v^L, \{S\}$)                          ▷ $\{S\}$: Set of SyncInfo

procedure AddConstraint($C_{new}$)
    if $C_{new}$ is a top constraint then
        $\mathcal{C}_{TOP} \leftarrow \mathcal{C}_{TOP} \cup C_{new}$     ▷ $\mathcal{C}_{TOP}$: set of top constraints, sorted by localtime
        Convexify($C_{new}$)
        if $C_{new} \in \mathcal{C}_{TOP} \wedge C_{new}$ violates upper line then UpdateUpperLine
    else                                              ▷ $C_{new}$ is Bottom
        $\mathcal{C}_{BOT} \leftarrow \mathcal{C}_{BOT} \cup C_{new}$     ▷ $\mathcal{C}_{BOT}$: set of bottom constraints, sorted by localtime
        Convexify($C_{new}$)
        if $C_{new} \in \mathcal{C}_{BOT} \wedge C_{new}$ violates lower line then UpdateLowerLine

procedure Convexify($C_{new}$)
    Find leftmost $C_i$ s.t. $\{C_i, ..., C_{new}\}$ is non-convex
    Remove non-convex constraints between $C_i, C_{new}$
    Find rightmost $C_j$ s.t. $\{C_{new}, ..., C_j\}$ is non-convex
    Remove non-convex constraints between $C_{new}, C_j$

procedure UpdateUpperLine
    $l \leftarrow$ FindTangent
    if Slope($l$)$> 1 + \eta$ then $l \leftarrow$ FindTopSupport($1 + \eta$)

procedure UpdateLowerLine
    $l \leftarrow$ FindTangent
    if Slope($l$)$< 1 - \eta$ then $l \leftarrow$ FindBottomSupport($1 - \eta$)
```

Update Limiting Lines. The core of the procedures for updating upper and lower lines is FindTangent. In FindTangent we find a tangent of two convex polygons. Note that polygons are convex whenever FindTangent is called. A naive method mentioned in [6] is to try all possible pairs of vertices and check if the line intersects with the polygons. This takes $O(n^2)$ time in the worst case, but can be improved to $O(\log^2 n)$ by using nested binary search. Further, by using more sophisticated algorithms, it is improved to $O(\log n)$ time [11,17].

When the slope of the calculated limiting line is out of range of $[1-\eta, 1+\eta]$, we can replace the line with the slope $(1+\eta)$ (for upper line) or $(1-\eta)$ (for lower line). In this case the line is supported by one point instead of two. Searching the support is done efficiently by calculating the slope of each edge of the polygon, and since it is convex, the slope is monotonically increasing (for top polygon) or decreasing (for bottom polygon). By using binary search, it takes $O(\log n)$ time.

5 Implementation

We have implemented the proposed synchronization algorithm in TinyOS 2.1.1. Besides dealing with limitations in TinyOS programming, we needed to address several fundamental issues specific to our synchronization algorithm.

5.1 Delay Compensation

Reducing the transmission delay between sender and receiver is the key to achieving precise synchronization. This also applies to our algorithm, as the minimum possible gap between upper and lower limits is at least twice as big as the minimum transmission delay (proof omitted).

To minimize the transmission delay, we use MAC-layer timestamping, which is to put a timestamp when the packet is actually transmitted over radio. In case of CC2420 radio chip, the timestamp for a packet is obtained when SFD (start frame delimiter) is captured. Since this is same for both sender and receiver, two timestamps are expected to be obtained at very close time points.

A problem in MAC-layer timestamping is the delay after a node calculates the lower limit until the packet is transmitted. A receiver can get a tighter bottom constraint if the lower limit is calculated at the time when the packet is transmitted. However, it is not feasible to recalculate it at MAC layer.

A solution for this problem is to reconstruct the lower limit at the receiver side. At localtime s_1, the sender calculates the lower limit $f^L(s_1)$, puts it in the packet, and issues the send command. In the timestamp field, which will be rewritten at the MAC-layer, the sender put s_1, the localtime when it calculated the lower limit. At the MAC-layer this field is rewritten with the difference with value in the field and the current timestamp s_2. When the packet is received, the receiver carries $f^L(s_1)$ and $\Delta s_{12} = s_2 - s_1$. Then it calculates the delay-compensated lower bound $\tilde{f}^L(s_2)$ as follows:

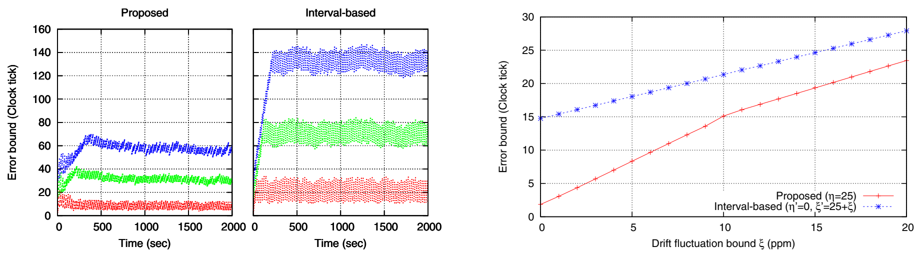
$$\tilde{f}^L(s_2) = f^L(s_1) + (1 - 3\eta - \xi)\Delta s_{12}.$$

This is based on the following theorem. Proof is omitted from this version.

Theorem 2. *$f^L(s_2) \geq f^L(s_1) + (1 - 3\eta - \xi)\Delta s_{12}$, where $f^L(s_2)$ is actual lower limit at localtime s_2.*

5.2 Quantization Error

There are several issues related to quantization error that affect the correctness of the algorithm. One is about the order of timestamping. We strictly require that the timestamp at the sender is taken before that at the receiver. However, when we consider integer timestamps, this is not always satisfied. This order violation will not happen if the receiver timestamp is always taken more than one clock tick after the sender's one, but unfortunately in our case, the difference is much smaller than one tick. To avoid this situation, we add one clock tick to the receiver's timestamp.



(a) Proposed vs. Interval-based method (b) Average error bound at first hop

Fig. 6. Comparison between proposed and interval-based methods

The other is about compensated constraints. Even though the original value of each constraint is an integer, it will become a noninteger when we calculate the compensated value. For not to violate the compensated constraints, we need to be conservative on quantization. Specifically, we round up the value for a top constraint and round down the value for a bottom constraint. Because of this, there are some cases that message transmission loops in two nodes without receiving any new messages. To avoid this, a node does not send a message if it has sent the previous one within a short ($\sim$ 1 sec) period.

6 Evaluations

In this section we evaluate the proposed synchronization algorithm by both simulation experiment and testbed experiment. For the performance metric, we use the error bound, which is calculated from the upper and lower limits by $(f^H(s) - f^L(s))/2$. This is the minimum error bound, which is achieved by using the average of upper and lower limits as the estimate.

6.1 Comparison with Interval-Based Method

First we compare the proposed algorithm and interval-based methods by simulation. We use a topology of one root at the end and 10 nodes connected in line. Figure 6(a) compares the error bounds of three nodes (at 1st, 5th, and 10th hop from the root) for both algorithms. For all three nodes, the proposed algorithm achieves much smaller error bounds than the interval-based method. This suggests the information on drift fluctuation bound can help improving the accuracy in clock synchronization.

To see this effect in more quantitative way, we change the drift fluctuation bound ξ and see the error bound. For the proposed method, the drift offset bound η is kept constant at 25ppm, and we emulate an interval-based method by setting $\eta' = 0$ and $\xi' = \eta + \xi$ based on the discussion in Section 3.4. Figure 6(b) shows the results for the first hop. The results show that the proposed method can achieve smaller error bound for all cases and the improvement is bigger for the smaller ξ.

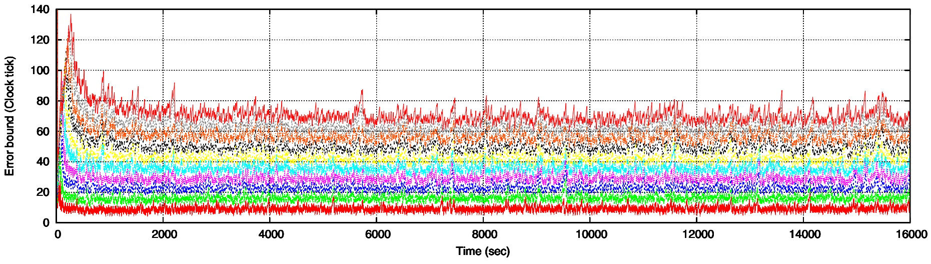
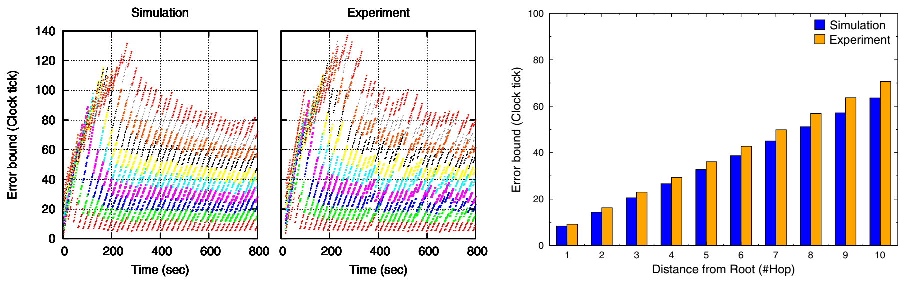


Fig. 7. Testbed experiment: Error bounds of 10 nodes in a line topology



(a) First 800 sec (b) Average after 800 sec

Fig. 8. Comparison between simulation and testbed experiment

6.2 Testbed Experiments

We programmed 11 Telos B nodes; one for root and other 10 for nodes. In the software we made a line topology with a root at one end. The UserINT ports of the root and all nodes are connected to a single trigger source that emits a trigger every two seconds. Upon receiving a trigger, the root node sends the current time and all other nodes send the upper and lower limits at that time to the PC. We use serial communication via USB port for the communication with the PC to avoid congesting the radio channel.

Comparison with Simulation Results. Figure 7 shows the error bound of all 10 nodes (except the root). Figure 8 shows the comparison between measured and simulated results. The change of error bounds closely matches the simulation result both in the transient state at first (Fig. 8(a)) and in the steady state after long time (Fig. 8(b)), though the average error bound in the experiment is a little (9-13%) higher than in the simulation. The error bound at the first hop is 9.2 ticks, which corresponds to 280.0 μsecs at 32768.5Hz clock, and it is roughly proportional to the hop distance from the root.

Improved Clock Estimation and Comparison with FTSP. As an extension we have incorporated the ideas from previous work to improve the clock estimation performance. We slightly change the proposed algorithm and add the current estimate of globaltime in each message, in addition to the lower bound of it.

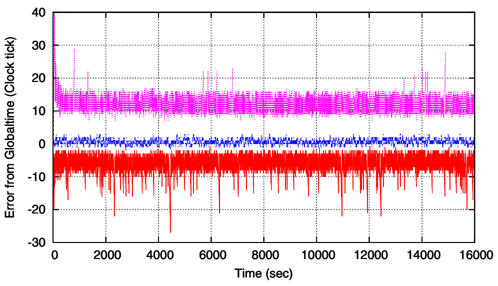


Fig. 9. Upper/lower limits and estimate at the first hop

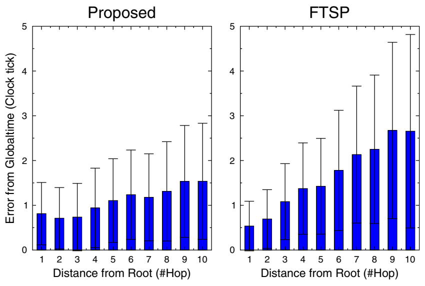


Fig. 10. Comparison with FTSP: Average estimation error

For estimation, we use a common method used in e.g.,[12,14]: Each node considers the time lapse between receiving and sending, makes a table of (localtime, globaltime) pair, and uses linear regression to obtain an estimate.

Figure 9 shows upper/lower limits and estimate at the first hop in the 10-node line topology. The vertical axis is the error from globaltime. Since the upper and lower limits are deterministic bounds, the error for upper limit is always positive and that for lower limit is negative. The estimate distributes around zero. Figure 10 shows the average estimation error for each of 10 hops line topology compared between the proposed algorithm and FTSP. It is observed that the average error for the proposed algorithm is comparable to that for FTSP in the near distance and up to 40% better in the far (At 10th hop, 1.54 vs. 2.65 ticks; 42% better).

7 Related Work

Our synchronization algorithm is most closely related to interval-based methods, as we have discussed in Section 3.4. Marzullo and Owicki [16] studied the synchronization problem where each time server returns an interval that contains the true time and the objective is to minimize the length of interval by exchanging messages among multiple time servers. Blum et al. [7] modified their algorithm for sensor networks and improved the average case performance. Römer proposed another interval-based algorithm tailored for ad hoc networks settings [18]. Schmid and Schossmaier [20] provided bounds under the assumption that several performance parameters such as transmission delay bounds as well as drift bounds are given. In our work, we have made our first attempt to give bounds only from the information readily available in the specifications of the hardware components. Potentially we can improve the results when we have more information about the hardware and the environment.

The use of convex polygons and estimation of clock functions as tangents to them was first proposed by Duda et al. [9] and studied in further detail by Berthaud [6]. In [23], the authors designed optimal and approximate algorithms to keep only the constraints that become supports. All these works assume constant drift either indefinitely or for a short time. Our algorithm share the same idea for determining the limiting lines, but without constant drift assumptions.

The notion of delay compensation (Section 5.1) has been discussed in [13,20]. We have extended these results under the bounded drift fluctuation assumption and also devised a method suitable for the case of MAC-layer timestamping.

Since MAC-layer timestamping reduces the transmission delay down to few microseconds, clock drift is the main issue for synchronization, since many sensor nodes use normal crystal oscillators with large drift rather than TCXO or OCXO due to energy and cost limitations. FTSP [15] estimates the clock drift and offset with high precision by using linear regression, assuming constant drift for a short period. GTSP [21] focuses on minimizing the local error among the neighboring nodes by adjusting the logical clock rate based on the estimation of clock drift of neighborhood nodes. PulseSync [14] improves the performance at distant hops by introducing a pulse-like propagation pattern. The latter two papers include convergence and optimality results on the performance including accuracy, but they are either asymptotic or probabilistic under the assumption of constant drift and bounded transmission delay. In this paper we focused on the deterministic accuracy guarantee instead, though we have demonstrated that the proposed algorithm can be combined with them to obtain good estimation.

Schmid et al. [19] proposed a method for compensating for the drift change due to temperature change. This is essentially a software implementation of what TCXOs do. Using the onboard temperature sensor, each node makes a table storing the pair of temperature and relative drift. After getting enough entries in the table, a node can estimate the drift with high accuracy and thus can compensate for that without communicating with other nodes. Our algorithm is orthogonal to this. As we have seen in Fig. 6(b), we can achieve tighter error bound for smaller drift fluctuation bound. Therefore, with this or any other techniques that reduce drift fluctuation and provide deterministic guarantee for that, we can improve the accuracy guarantee with the proposed algorithm.

8 Conclusions

We have presented a clock synchronization algorithm that gives deterministic accuracy guarantees. The main idea is to find the upper and lower limits of clock function that satisfies all the constraints obtained using causality relations in communication. While the idea is similar to classical interval-based synchronization methods, we do not make simplifying assumptions such as constant drift or negligible transmission delay to obtain strict guarantees. Still, as demonstrated by the experiments, we achieve tighter guarantees owing to the bounded drift fluctuation assumption, which is confirmed by the hardware specification as well as by the preliminary experiments. We have implemented the algorithm in TinyOS and demonstrated that the accuracy guarantees are close to the simulation results. Furthermore, we extended the algorithm to obtain good estimation and achieved the estimation error up to 40% better than FTSP.

Acknowledgments. This work was supported in part by UCSD/LANL Engineering Institute and by NSF under grant numbers CNS-0932360, CCF-0702792, and SRS-0820034. The first author thanks Eric Yip for his help on the TinyOS implementation. The first author also thanks Muhammad Abdullah Adnan and Thomas Schmid for fruitful discussions.

References

1. CMR200T data sheet, `http://www.citizencrystal.com/images/pdf/k-cmr.pdf`
2. GPS 18x Technical Specifications, `http://www8.garmin.com/manuals/GPS18x_TechnicalSpecifications.pdf`
3. TinyOS Hardware Designs: Telos (Rev B) BOM, `http://webs.cs.berkeley.edu/tos/hardware/telos/telos-revb-bom-2004-09-21.xls`
4. MSP430 32-kHz Crystal Oscillators, Rev. B (2006), `http://focus.ti.com/lit/an/slaa322b/slaa322b.pdf`
5. MSP430x1xx Family User's Guide, Rev. F (2006), `http://focus.ti.com/lit/ug/slau049f/slau049f.pdf`
6. Berthaud, J.M.: Time synchronization over networks using convex closures. IEEE/ACM Trans. Networking 8(2), 265–277 (2000)
7. Blum, P., Meier, L., Thiele, L.: Improved interval-based clock synchronization in sensor networks. In: IPSN (2004)
8. Capkun, S., Cagalj, M., Srivastava, M.: Secure localization with hidden and mobile base stations. In: INFOCOM (2006)
9. Duda, A., Harrus, G., Haddad, Y., Bernard, G.: Estimating global time in distributed systems. In: ICDCS (1987), `http://scholar.google.com/scholar?hl=en\&btnG=Search\&q=intitle:Estimating+Global+Time+in+Distributed+Systems#0`
10. Graham, R.L.: An efficient algorithm for determining the convex hull of a finite planar set. Information Processing Letters 1, 132–133 (1972)
11. Kirkpatrick, D.G., Snoeyink, J.: Computing common tangents without a separating line. In: Sack, J.-R., Akl, S.G., Dehne, F., Santoro, N. (eds.) WADS 1995. LNCS, vol. 955, pp.183–193. Springer, Heidelberg (1995)
12. Kusy, B., Dutta, P., Levis, P., Maroti, M., Ledeczi, A., Culler, D.: Elapsed time on arrival: a simple and versatile primitive for canonical time synchronisation services. Int. J. Ad Hoc Ubiquitous Comput. 1(4), 239–251 (2006)
13. Lamport, L.: Synchronizing time servers. SRC Research Report 18 (1987)
14. Lenzen, C., Sommer, P., Wattenhofer, R.: Optimal clock synchronization in networks. In: SenSys (2009)
15. Maróti, M., Kusy, B., Simon, G., Lédeczi, A.: The flooding time synchronization protocol. In: SenSys (2004)
16. Marzullo, K., Owicki, S.: Maintaining the time in a distributed system. In: PODC (1983)
17. Overmars, M.H., van Leeuwen, J.: Maintenance of configurations in the plane. J. Comput. Syst. Sci. 23(2), 166–204 (1981)
18. Römer, K.: Time synchronization in ad hoc networks. In: MobiHoc (2001)
19. Schmid, T., Charbiwala, Z., Shea, R., Srivastava, M.B.: Temperature compensated time synchronization. IEEE Embedded Systems Letters 1(2), 37–41 (2009)

20. Schmid, U., Schossmaier, K.: Interval-based clock synchronization. Real-Time Systems 12(2), 173–228 (1997)
21. Sommer, P., Wattenhofer, R.: Gradient clock synchronization in wireless sensor networks. In: IPSN (2009)
22. Vig, J.R.: Introduction to quartz frequency standards. Tech. Rep. SLCET–TR–92–1, Army Research Laboratory (1992), `http://www.ieee-uffc.org/frequency_control/teaching.asp?name=vigtoc`
23. Yoon, S., Veerarittiphan, C., Sichitiu, M.L.: Tiny-sync: Tight time synchronization for wireless sensor networks. ACM Trans. Sensor Networks 3(2), 8 (2007)

A Two-Way Time of Flight Ranging Scheme for Wireless Sensor Networks

Evangelos B. Mazomenos, Dirk De Jager, Jeffrey S. Reeve, and Neil M. White

Electronic Systems and Devices Group,
School of Electronics and Computer Science,
University of Southampton, SO17 1BJ, UK
{ebm07r,ddj07r,jsr,nmw}@ecs.soton.ac.uk

Abstract. Relative ranging between Wireless Sensor Network (WSN) nodes is considered to be an important requirement for a number of distributed applications. This paper focuses on a two-way, time of flight (ToF) technique which achieves good accuracy in estimating the point-to-point distance between two wireless nodes. The underlying idea is to utilize a two-way time transfer approach in order to avoid the need for clock synchronization between the participating wireless nodes. Moreover, by employing multiple ToF measurements, sub-clock resolution is achieved. A calibration stage is used to estimate the various delays that occur during a message exchange and require subtraction from the initial timed value. The calculation of the range between the nodes takes place on-node making the proposed scheme suitable for distributed systems. Care has been taken to exclude the erroneous readings from the set of measurements that are used in the estimation of the desired range. The two-way ToF technique has been implemented on commercial off-the-self (COTS) devices without the need for additional hardware. The system has been deployed in various experimental locations both indoors and outdoors and the obtained results reveal that accuracy between 1 m RMS and 2.5 m RMS in line-of-sight conditions over a 42 m range can be achieved.

1 Introduction

The ability to estimate the relative distance between low-power wireless embedded nodes is paramount for a number of applications which require location-awareness [5,13]. In the general case, two or more nodes will engage in some kind of interaction, typically transmit and/or receive signals, and will be tasked with measuring a property of the signal that can be appropriately processed in order to extract the relative distance between the two interacting nodes. For example, by measuring the Received Signal Strength (RSS) value of a signal, the range between the two nodes can be derived [1]. Another well-known approach is based on calculating the transit time of a signal and use it to estimate the point-to-point range of two nodes. These methods are known as Time of Flight (ToF) or Time of Arrival (ToA). The amount of time that a signal requires to reach the receiver is measured with the use of on-node clocks. The a-priori knowledge of the signal's velocity enables the approximation of the desired distance.

Important advancements in microelectronics technology over the past decade, resulted in the production of very accurate clocks (*ns* accuracy) in electronic devices. As a result a variety of ToF methods has been utilized in a number of established navigational and positioning systems (e.g. GPS). Consequently, as node localization became a necessity in WSNs, ToF techniques for low-power sensor nodes have been investigated. One major area of concern, has been the fact that low-power embedded devices are not equipped with high frequency clocks and time synchronization in these devices is an inherently difficult task, which also has attracted significant research interest [9, 17].

In this paper a two-way ToF ranging technique for WSNs is proposed. Our approach is to employ a two-way ToF method in order to avoid the need for synchronization between the participating nodes. This method targets low-power embedded devices, thus the clocks that are considered, operate at relatively low-frequencies (up to 32MHz). Multiple two-way ToF measurements are obtained which allows the system to achieve sub-clock resolution. The final ToF value is extracted after averaging the accumulated timing values. A simple, yet practical procedure is employed to eliminate any erroneous data which are caused because of surrounding noise or sampling artifacts. A calibration step is also carried out to exclude the delays that are included in the two-way path of the signal. Accuracy and latency are important in a ranging system for low-power embedded nodes capable of operating in real-time. Different to a number of approaches in this research area where the accumulated data require significant post-processing, the proposed system completes all the necessary processing on the nodes that participate in the ranging operation. The range estimate can then be exploited according to the application needs. The key contributions of this paper are the implementation of a two-way ToF ranging method on COTS hardware and the evaluation of its performance on real-world experiments.

The remainder of this paper is organized as follows. The following section performs a review of previously proposed ToF systems in WSNs with some background information regarding ToF ranging. In sequel, the specific details of the two-way ToF ranging that is proposed, are provided in Section 3 alongside an investigation of the error sources. A thorough analysis, of the implementation on hardware follows in Section 4. Section 5 presents the experiments that were carried out in indoors and outdoors locations and the analysis of the results obtained. Finally, Section 6 summarizes the key points and concludes the paper.

2 Related Work

ToF ranging systems attempt to estimate the point-to-point distance between two communicating devices by capturing the time that a signal requires to travel from one device to the other. Since the speed of the signal is known and constant (e.g., the speed of light for electromagnetic signals), the distance can then be calculated. McCrady *et.al* are among the first to propose a ToF ranging system for WSNs [12]. However their work lacks implementation. The RSSI and

ToF methods have been combined in a locationing system [14]. Ultra Wideband (UWB) transceivers have the ability to yield fine-grained resolution in measuring the ToF due to the high bandwidth occupancy. Thus, a handful of ToF ranging systems are based on Ultra Wideband (UWB) technology [2,3,8]. However, low-power WSNs nodes are normally not equipped with UWB transceivers and their incorporation on embedded nodes presents a number of challenges. Lanzisera *et. al.* propose a ToA locationing scheme for low-power ASIC WSN nodes [7]. In the prototype an FPGA board is attached to the WSN node to carry out the necessary calculations. FPGA boards alongside WSN nodes are also used in [6], where a RF-ToF ranging system is presented and the ToF is extracted by the channel impulse which is produced after converting the received signal from the time to the frequency domain by applying FFT. For this procedure, both FPGA and DAC are used. The approach we propose, differentiates from the previous approaches since it does not require any additional per-node hardware.

An intriguing approach for ToF ranging in WSNs is the one that employs acoustic signals instead of electromagnetic ones. It is known that acoustic signals travel in a much slower speed than electromagnetic signals thus making them easier to utilize in ToF scenarios. Both ultrasonic and audible sound signals have been utilized in ToF ranging systems. Occasionally, acoustic and RF signals can be combined in a time difference of arrival method (TDoA). The two signals are emitted simultaneously and the RF signal is used to synchronize the receiver. The TDoA value is considered to be the ToF of the acoustic signal. A ranging system based on this approach is implemented on the Mica2 mote in [16]. A simple tone which is produced by the mote's sounder is the acoustic signal that it is timed. The "Calamari" localization system follows a similar approach but employs the tone detector of the Mica mote instead of the sounder and requires all participating nodes to be pre-calibrated to achieve good accuracy [23]. Another example where acoustic and RF signals are used on the same system is the "Cricket" locationing system developed at MIT [15]. One disadvantage of acoustic ranging, is the limited effective range of acoustic-based ranging systems. The systems presented previously are capable of producing accurate ranging but within a limited range. Radio interferometric geolocation is another method to estimate the range in wireless embedded nodes, by using the radio interferometry principle. According to the authors in [10] very good accuracy ($< 10cm$) can be achieved. The major drawback of this system, which makes it unsuitable for real-time ranging, is that a significant amount of time is required for the ranging algorithm to run to completion.

The proposed ranging method for low-power embedded nodes is inspired by the work presented by Thorbjornsen *et. al.* in [22]. Our intention is to evaluate the two-way ToF ranging technique and ultimately incorporate it in the range-only tracking system presented in [11]. The approach presented here, attempts to achieve better resolution in timing the value of the two-way message exchange by employing a different method on how the timer's value is captured. Instead of detecting a received message by sampling the receiver with a constant sampling rate, the receiver is programmed to signal an interrupt whenever a ranging message has completed a two-way path. The interrupt routine is then used to capture the value of the running timer. This approach results in better resolution of the two-way timing values, thus achieves better resolution in the resulting distance by processing multiple two-way transactions between the participating nodes.

3 Overview of the Proposed System

The basic concept of the proposed two-way ToF ranging system is illustrated in Fig. 1. The objective is to estimate the distance between node A and node B.Initially node A sends the first ranging signal and captures the time of its timer ($\mathbf{t}_{tAB}$). Node B receives the signal and after a period of time, that corresponds to node B swapping its state, from receiver to transmitter (as well as a number of other delays) node B sends a ranging signal back to node A. Following, node A receives the reply signal and stores the time of reception ($\mathbf{t}_{rBA}$). The timer in node A measures $\mathbf{t}_A = \mathbf{t}_{rBA} - \mathbf{t}_{tAB}$ multiple times. Instead of using a clock at node B to measure the time that the signal spends in the node, our approach is to measure all the delays that occur during this two-way signal exchange process. This is accomplished by placing the transceivers at a minimum distance ($< 0.2m$) and executing multiple transactions that are averaged to produce the minimum time($\mathbf{t}_{min}$) that is required in order to complete a message exchange. This time corresponds to a minimal ToF period and reveals all the hardware and software delays that occur during a two-way ranging transaction. We make the assumption that the these delays remain constant and are independent of the distance between the nodes. Subsequently only the propagation delay will increase the two-way time transfer value as the nodes are placed at greater distance.

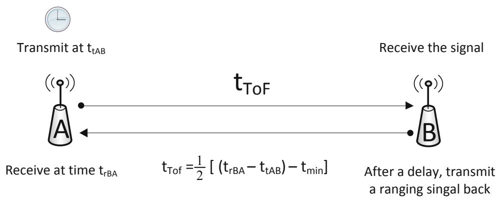

Transmit at t_{tAB}
Receive the signal
t_{ToF}
A
B
Receive at time t_{rBA}
$t_{Tof} = \frac{1}{2}[(t_{rBA} - t_{tAB}) - t_{min}]$
After a delay, transmit a ranging singal back

Fig. 1. Proposed Two-way ToF Ranging

Figure 2 illustrates a timing diagram of a message exchange between the two nodes. Send and receive occurs on the rising edge of the nodes clocks. Assuming that for a set distance the t_{ToF} will be the same and the delay T_{B_proc} that node B requires to process the ranging signal and submit the reply is constant, then the only ambiguity will be inserted by the delays associated to the clocks phase shift and frequency drift. Given that the two clocks are unsynchronized and have a small difference in frequency the phase offset between the devices will oscillate, thus the delays T_{d1} and T_{d2} will follow a similar varying pattern. By oversampling, we capture a normally distributed set of multiple timing transactions centered around the mean ToF value. Subsequently, capturing a sufficiently large number of timing values will allow us to extract the mean ToF value from the Gaussian distribution which can be, linearly associated to the distance between the nodes.

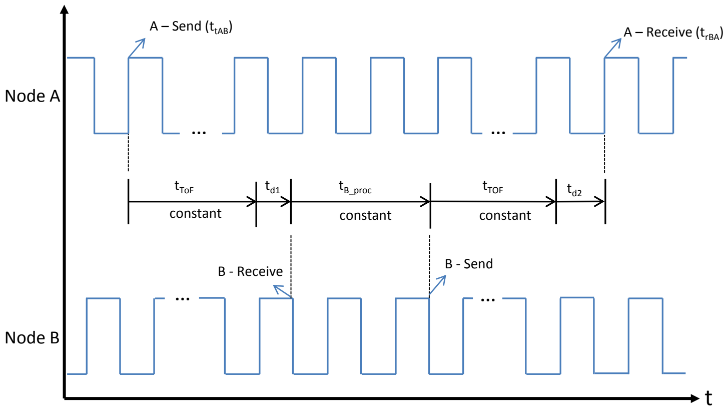


Fig. 2. Timing Diagram of a two-way message exchange

The calculation phase involves the extraction of the ToF out of the multiple stored timer values. In the event that one, or in general a small fraction of these n transactions has produced erroneous timing, including them in the average calculation will result in a distortion of the correct mean value. To avoid this, and to assure that the ToF calculation is based on the most accurate and "true" transactions the following procedure is followed. Let us assume that we obtain n two-way ToF values $\mathbf{t}_n$. The initial average mean $\widetilde{\mathbf{t}}_{ToF}$ and the standard deviation of the n values is given from the following:

$$\widetilde{\mathbf{t}}_{ToF} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{t}_n \qquad \sigma_{ToF} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{t_n} - \widetilde{\mathbf{t}}_{ToF}) \tag{1}$$

In the following step we calculate the absolute difference of each one of the n values from the initial mean. Ultimately, out of the n collected ToF values we exclude the ones that their absolute difference to the initial mean is greater than the standard deviation. The final $\widehat{\mathbf{t}}_{ToF}$ value is calculated by averaging the remaining m values.

$$\widehat{\mathbf{t}}_{ToF} = \frac{1}{m}\sum_{i=1}^{n}\mathbf{t}_m \tag{2}$$

Obtaining the two-way ToF is the main step in estimating the range between the two nodes. That value is converted to distance by executing the following.

1. Calibrate the $\widehat{\mathbf{t}}_{ToF}$ value by subtracting it from the minimum two-way ToF. The minimum two-way ToF ($\mathbf{t}_{min}$) is obtained by placing the nodes at a minimum distance and averaging n transactions in a similar way as mentioned previously.
2. Divide the calibrated value by two, to get a single-way ToF time. $\mathbf{t}_{ToF_{final}} = \frac{1}{2}(\widehat{\mathbf{t}}_{ToF} - \mathbf{t}_{min})$.
3. Multiply the above with the speed of light to convert time to distance

3.1 Sources of Ranging Error

The achievable accuracy of any RF-ToF ranging system is primarily limited by several factors which introduce temporally and spatially random errors [4]. Noise as well as interference, are two major factors that can cause the accuracy to degrade. For example, noise can cause the receiver to detect signals in the wrong time leading to faulty measurements. The effect of noise in RF-ranging methods can be quantified with the use of the Signal-To-Noise Ratio (SNR) on the receiver's side and the occupied bandwidth (B). These measures are linked via the Cramér-Rao Lower bound (CRB). For two-way ToF ranging systems and n measurements averaged, the CRB of variance σ^2_{ToF} is given by the following relationship [19].

$$\sigma^2_{ToF} \geq \frac{c^2}{2(2\pi B)^2 \cdot SNR \cdot n} \tag{3}$$

Clock synchronization is a key aspect in every ToA system. The times of transmission and reception of wireless signals must be known using a common time base in order to deduce accurate measurements. Clock synchronization is of particular importance in one-way ToA methods. Two-way methods exhibit an advantage over the one-way method since each node operates its own clock, hence its own timing system. Nevertheless, in order to extract the ToF value in a two-way ranging method, the delay time in the replying node as well as the offset between the node clocks must be approximated and then taken into account in the calculation of the ToF value. ToF systems can also be affected from multipath propagation. Multipath interference typically occurs, because the transmitted signal bounces off objects in the environment, and then adds to the LoS signal. Consequently, the LoS signal can be severely attenuated which may result in the signal being incorrectly received or lost completely. The error caused by multipath interference is difficult to be quantified as it depends upon the deployment environment.

Apart from the sources of ranging error that were analysed previously a number of additional uncertainties may add non-deterministic delays that will result in distorted timing of the two-way round trip timing value. A thorough analysis of these uncertainties is performed by Maróti *et. al* [9] in their work on synchronization techniques. One must also consider that an additional factor of uncertainty will be the drift over time that the clock oscillator on the embedded node will demonstrate. The output frequency of the node's clock is susceptible to drift and is affected by the surrounding temperature and the node's supply voltage. It is therefore, not uncommon to observe different latencies even on the same hardware. Additional timing uncertainties may incur from the node's radio operation during the submission and reception of packets. These uncertainties are influenced by factors such as the message length, the interrupt handling and channel availability. It is imperative to ensure that the effect of these errors will remain constant as possible in the implementation of the proposed ToF system in order to avoid erroneous timing of the two-way message transmission that will result in diminishing ranging accuracy.

Due to the previously mentioned reasons, we expect the ToF values to vary for a set distance. During the calibration stage the additional delays introduced by these factors must be sufficiently captured in order to be excluded from the ToF values. To achieve this, the combined delays which are introduced by these factors must remain as constant as possible during any experimental set-up. By oversampling the ToF values sufficiently the errors that are associate to the timing uncertainties are averaged out and do not affect the mean calculated averaged measurements given that the calibration values is removed.

4 Implementation

The Texas Instruments (TI) EZ430-RF2500 is a complete wireless development platform which combines the MSP430 microcontroller (MCU) and the CC2500 low-power radio module. The EZ430-RF2500 target board connects to a standard USB port for programming, debugging and communications purposes. The MCU is a 16-bit microcontroller with a clock speed up to 16MHz. It employs 32kB of flash memory and 1 kB of RAM. The clock system includes a low-power 12KHz crystal oscillator (VLO) and a more accurate and energy demanding digital controlled oscillator(DCO) which can be set to a range of frequencies (1-16Mhz). The DCO operates on factory calibrated settings that demonstrate improved tolerance against temperature and voltage supply compared to previous versions of the MSP430 family of microcontrollers. Two timers/counters (named Timer A (16bit) and Timer B (variable bit-length)) are present and can be linked to the clock sources [20]. The CC2500 is a low-cost radio transceiver operating in the 2.4GHz RF band, designed for low-power embedded applications. Various modulation formats (OOK, 2-FSK, MSK) and data rates (2.4 - 500 kBaud) are supported. The configuration of CC2500 is done by programming 8-bit registers. Three registers are associated to general purpose output digital pins (GDO0-2) that can be used in various ways. The CC2500 radio does not directly support the IEEE 802.15.4 frame format. It uses a proprietary format, similar to the one defined in the 802.15.4 protocol. The CC2500 consists of a variable length preamble sequence, a synchronization word (SYNC WORD), a length byte, an address byte, the data payload and finally an optional two-byte cycle redundancy check field (CRC). The packet's maximum length is 256 bytes [21]. For the purposes of the proposed ranging system the following configuration settings were made for the CC2500 transceiver. Two data rate settings were used at

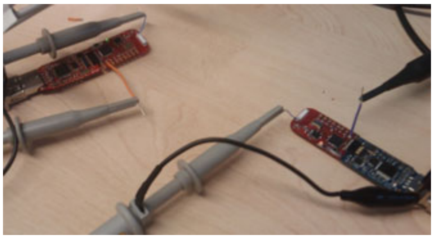
250kbps and at 500kbps. The transmission power was set to the highest value possible, that is +1dBm. The modulation used is minimum-shift keying (MSK), the preamble length was set to 2 bytes and the SYNC WORD to 4 bytes. Finally SYNC WORD detection was set to 30/32 bits.

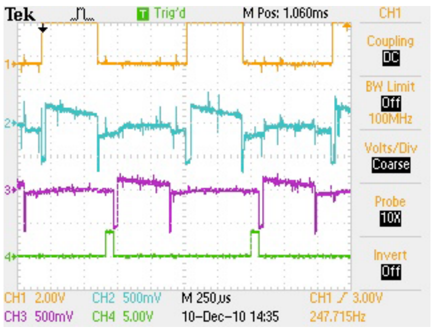
To achieve the maximum possible resolution in timing the two-way ToF value, Timer A is set to continuously count-up mode and is sourced to the DCO which is set at the maximum possible clock frequency at 16 MHz. In order to capture the ToF a GDO pin is configured to change state to "high" whenever a SYNC WORD is transmitted or received. The GDO pin returns in low "state" when the entire packet is transmitted/received. In the developed software, the GDO pin is programmed to trigger an interrupt in the event that it changes state from low-to-high. Using that interrupt, Timer A resets whenever a SYNC WORD of a ranging message is transmitted and its value, which correspond to the two-way ToF, is captured directly from the hardware register only when a SYNC WORD is received (assuming that the incoming message is transmitted from the other device that takes place in the ranging procedure) through the same interrupt routine. A binary variable acts as a lock in order to avoid unwanted capturing of the timer's value. This method avoids the need for sampling the pin with a predefined rate, since the GDO pin itself triggers the interrupt and offers better resolution.

As mentioned earlier, the two-way ToF ranging is performed between a pair of EZ430-RF2500 devices programmed independently with different software. One of them is termed as the *requester* and the other one as the *responder*. The *requester* is the device that initiates the sequence in order for the two devices to engage in exchanging the necessary ranging messages. Practically, the *requester* device controls the initiation and termination of the ranging process. The software that we developed was designed with the following in mind. Both the *requester* as well as the *responder* code must be as simple as possible. No additional interrupts or unnecessary operation should intervene during the transmission of the ranging packets. The packets to be sent, should be in terms of size, as minimum as possible to eliminate any delays on the MCU and radio load. Our major goal is to maintain a constant delay primarily in the *responder* node during the relay of the ranging packets. Of all the sources of delay mentioned analyzed by Maróti *et. al* we try to keep the propagation delay as the primary source of variability in the timing of the two-way ToF value. In conclusion, the developed software targeted at maintaining the hardware delays as constant as possible.

Additional precautions were taken to minimize the effect of uncertainty sources during a single two-way transaction. As pointed out in [9], various factors affect the uncertainties in a message transmission. Through our implementation we tried to maintain these uncertainties as constant as possible. Constant packet length was used to avoid varying transmission/reception times. The clear channel assessment option was not used as we assumed that there was no contention in accessing the channel during the experiments. An important source of delay that we had to tackle, is the amount of time the *responder* needs to acknowledge a correct ranging signal and reply accordingly. To guarantee a constant response time on the *responder's* side, we used a minimal static code routine specifically for this application. All other interrupts were disabled apart from the one associated to message detection. To evaluate the *responder's* reply delay, we used the same 16MHz clock to capture the time from the moment a packet is detected at the *responder* until the reply message is transmitted back. This delay, which includes the $9.6\mu s$ that is required for the transceiver to change state from Rx to Tx, was found to be constant during the exchange of ranging signals. Nevertheless one of the latencies that we were not able to address pertains to the interrupt handling. In essence, we assume that the moment an interrupt flag is raised, from the radio to signal the reception of a ranging packet, the MCU starts responding to that interrupt accordingly. However in reality there might be a sub-clock delay between the signalling of the interrupt by the radio and the MCU's response, due to the fact that the two components operate on different clocks. An approach that could mitigate these effects is to drive both the MCU and Radio from the same clock source. To evaluate the delays associated with the timing between the nodes, the code on the *requester* node was altered to set a pin high immediately after the Send Packet command was strobed to the CC2500 Radio, and the code on the *responder* was altered to set a pin "high" immediately when a packet was received. Two small connections where then soldered to the transmitting and receiving antennas of the devices, and a Teltronix TDS2014 Four Channel Oscilloscope was then connected to the transmitting and receiving nodes. This is visualized in Fig. 3(a). In Fig. 3(b), Channel 1 and 2 of the Oscilloscope represent the MCU pin set "high" immediately after the transmit packet command was strobed; and the signal transmitted on the Antenna respectively. Channel 3 and 4 of the Oscilloscope represent the *responder's* antenna signal and the pin set high on successful reception of a packet. From the timing analysis it can be seen that the transmit to receive signal on the MCU takes approximately 520μs (at 250kbps) which corresponds to 8320 counts of an accurate 16MHz clock. This means that our timing values on the *requester* correspond well to the total time measured by the oscilloscope. We thus assume that the radio operation does not add any significant uncertainty to our measurements and the timing values distribution is associated to the phase offset and clock drift (see section 5.2).

In the *requester* device a slow clock (12 KHz) sourced at Timer B, triggers the initiation of the entire process. When Timer B fires, the *requester* device sends a "request to send" packet and waits for the *responder* to reply . This procedure is repeated twice to ensure that the communication link between requester-responder is established successfully. Following, the requester begins the transmission of the first ranging packet (a simple packet with minimum payload) and also resets the value of the 16 MHz timer as explained previously (after the transmission of the SYNC WORD). Immediately after the transmission is completed, the requester switches its status to receiver and waits for a return packet from the *responder*. Upon, a successful reception of a ranging packet by the *responder*, the *responder* verifies that the received packet is a correct ranging


(a)



(b)

Fig. 3. Investigation of the timing uncertainties

packet (it also swaps status from receiver to transmitter) and then transmits back a ranging packet at the *requester*.

On reception of a ranging packet at the *requester* following previous transmission of a ranging packet, the GDO pin triggers an interrupt (when the SYNC WORD of the incoming packet is correctly detected) which captures Timer's A value which corresponds to the two-way ToF and the additional delays. This implementation with the use of an interrupt will yield better resolution than sampling the GDO pin with a constant sampling rate as in [22]. When the full packet is received, it is checked for correctness and if it is found to be correct the captured value is stored. The ranging transaction counter is incremented and the next cycle of ranging transmissions begins. This two-way packet exchange process is repeated until the nominal ranging transactions number is reached. The *requester* device then enters the calculation phase. In the event that a false packet has triggered the interrupt the captured value is considered erroneous and disregarded. The calculation phase was described in the previous section and pertains to the extraction of the ToF value. With this method the ToF averaged value is refined from all the values that might be erroneous. After the final ToF estimate is produced, the program resets all the variables and waits for Timer B to fire the next time in order for the same procedure to be repeated.

5 Deployment and Results

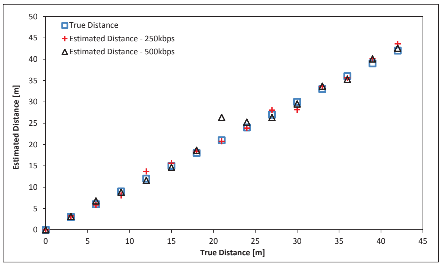
The two-way ToF method was tested on field experiments in order to evaluate the ranging precision and overall performance of the method. The experimental setup consisted of a pair of EZ430-RF2500 wireless nodes programmed with the *requester* and *responder* code respectively. The ideal environment for this type of experiment is an obstacle free area with good line-of-sight (LoS) for the two nodes. In addition, the interference from other wireless systems must be as low as possible. Since the CC2500 radio transceiver operates on the 2.4GHz band, it is expected that a number of other wireless networks, like WLAN, will cause significant interference if the deployment takes place in areas where such networks are present. Experiments were carried out on three different locations.

The first site (Location 1) is a level grass field at the University of Southampton campus where surrounding buildings could be a reason for multipath propagation and a number of WLAN university networks might cause interference. In this site the maximum communication range between the two nodes was limited due to space restrictions to 42m in LoS. In the second site (Location 2), nodes were deployed on a grass field with no obstacles being close to the experimental set-up. The maximum range between the two nodes that allowed the ranging system to run adequately was 70m in good LoS conditions. A number of experiments took place indoors in a narrow building corridor at the University of Southampton, School of Electronics and Computer Science (Location 3). The hallway is 42m long and had a maximum width of approximately 2m and minimum of 1.7m. In such an environment distortions in the measurements are expected due to multipath effects.

5.1 Experimental Setup

The EZ430-RF2500 devices were strapped on two wooden chairs to avoid the signal bouncing off the ground. The elevation was 90 cm off the ground. Both nodes were powered on from laptops to ensure that they operated with full power supply. The laptop on the *requester* was also used in order to log the ranging data via its USB port. The transmission power of the CC2500 radio was set at the maximum possible value of +1dBm. Two data rate settings were used for the node's radio in these experiments at 250kbps and at 500kbps. Due to the EZ430-RF2500 design, the antenna orientation plays a significant role on the maximum communication range. We concluded that the best antenna orientation was with the two antennas facing each other and being slightly inclined at an angle from the vertical position, towards the ground. Ranging data were collected from the *requester* node in steps of 3m until the maximum communication range where the experiment was adequately running was reached. A tape measure was used as reference and in order to measure the "true" distance between the two nodes. Initially the reference two-way ToF was estimated by placing the two nodes at a minimum distance ($< 0.3m$) and averaging 100 two-way transactions. In these experiments 1000 two-way transactions were used to estimate the distance between the two nodes. The calculation phase was executed every 100 transactions, thus 10 times in every location to reach the nominal 1000 values. The *requester* nodes was then moved to the next position. The metric used to evaluate the system's accuracy is the RMS error which is defined as follows. Assuming we estimate n positions: $d_{rms} = \sqrt{\frac{1}{n}(\mathbf{d}_{real} - \mathbf{d}_{esti})^2}$. It must be highlighted that within the purposes of this work we focused on the point-to-point range between two embedded nodes only. If multiple pair of nodes are to be engaged in ranging, the calibration stage must be executed for each individual pair of ranging nodes.

Results from two different days of experiments and for both data rate values are provided for Location 1 and Location 3. In Location 2 only the 250kbps was used as we tried to reach the maximum communication range and the slowest data rate facilitated our attempt. The collective results are illustrated in Figs. 4–5 and in Table 1.



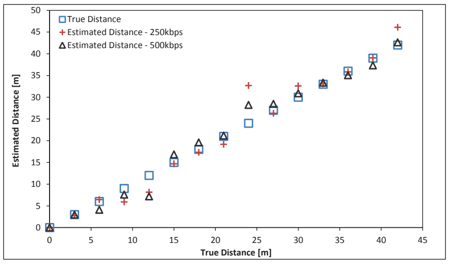


Fig. 4. Ranging results from experiments in (a) Location 1 and (b) Location 3

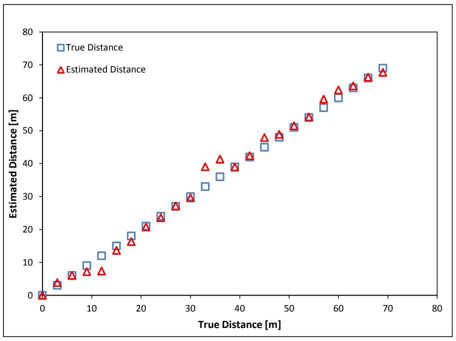


Fig. 5. Ranging results from Location 2

Table 1. Results from experiments for the proposed ToF method

Ranging Results			
Loc.	Data rate	RMS error	Max. Error
Loc. 1	250kbps	0.75m	1.79m
	250kbps	0.94m	1.84m
	500kbps	1.51m	5.32m
Loc. 2	250kbps	2.23m	6m
Loc. 3	250kbps	2.51m	5.32m
	500kbps	1.99m	4.82m

5.2 Performance Analysis

First of all, the timer that was used in the timing process is a 16MHz timer (maximum allowed value for the MCU). This value provides a resolution of $1/16MHz \times c = 18.75m$ In section 3.1 the CRB for a two-way ToF ranging method was formulated. At 250kbps the CC2500 transceiver occupies 540KHz of bandwidth while at 500kbps 812KHz. Considering the radio setting (output power +1dBm) and a typical environment where our experiments are conducted, a -5db SNR is an expected value. Hence from Equation 3 the lower bound of the variance of the proposed system, given that 1000 measurements are used, is $\sigma_{ToF}^2 = 137.3ns$ for the 250kbps and $60.7ns$ for the 500kbps respectively. These values correspond to a minimum ranging error standard deviation of 3.5159m and 2.33m for each data rate respectively (calculated from $\sigma_{ToF} \cdot c$). It must be noted that the -5db SNR is a typical value and in general the SNR varies in different deployments.

To measure the drift in clock frequency, we used a Hameg HM8123 frequency counter connected to a 10MHz SRS FS725 Rubidium Frequency Standard clock reference, and measured the clock frequency approximately every second over a period of 3 hours under room temperature and constant power supply. The HM8123 gating time was set to 100ms. We recorded the frequency from the HM8123 via a laptop's USB port. The results reveal that the DCO clock frequency is normally distributed with a standard deviation of 1.63KHz. The clock's accuracy is therefore in the area of 1% and the drift exhibits a standard deviation of 0.01% around the mean value. Due to this behavior, an additional error of around 17cm per clock cycle will be inserted because of the clock's instability. This frequency distribution yields a distribution of the time values similar to the one illustrated in Fig. 7.

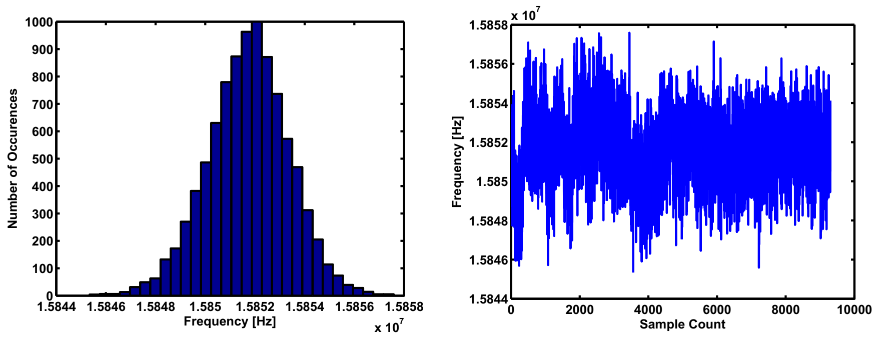
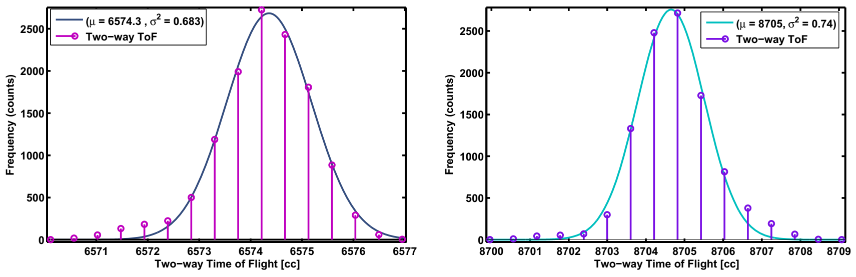


Fig. 6. Investigation of the 16Mhz clock inaccuracy. Frequency Histogram (a); Frequency vs Time (b)

As stated previously the node performed the necessary calculations whenever 100 two-way transactions were completed. Part of the process is the calculation of the standard deviation for these 100 transaction in order to exclude the timing values that fall outside the single deviant boundary. This procedure was repeated for 10 times in order to reach 1000 transactions. From all the experiments carried out the standard deviation of the timing values before averaging, was in the range of $1.4cc - 1.8cc$ for the 500kbps setting and $2.4cc - 3cc$ for the 250kbps. After averaging the 100 values the deviation was reduced to $0.4cc - 0.8cc$ for both the 500kbps and the 250kbps. Assuming a value of $0.6cc$ and dividing this by two we get $\sigma_{ToF} = 0.3cc$. This value is expressed in clock cycles (cc) and a single clock cycle of the 16MHz timer is $(1/16MHz = 62.5ns)$. Thus the standard deviation of the proposed system can be approximated as $\sigma_{ToF} = 18.75ns$. This translates to a maximum standard deviation of 5.62m. To sum up, given the theoretical derived lower bound, we presented a ToF ranging technique which exhibits a maximum standard deviation at the same order of magnitude as the theoretical calculated using the CRB lower bound.

To also verify the distribution of the measurements that the proposed ToF ranging system yields, an experiment is designed where two nodes are placed in a short distance ($\tilde{2}m$) and a vast number of ToF estimates is logged over a period of time. This experiment was executed with both data rate values. Approximately 10000 two-way ToF values were logged in each execution. The values are plotted according to 15 equally spaced bins. From Fig. 7 it is clear that the values can be considered as normally distributed and exhibit a standard deviation which is very close to the one observed in the previous experiments.

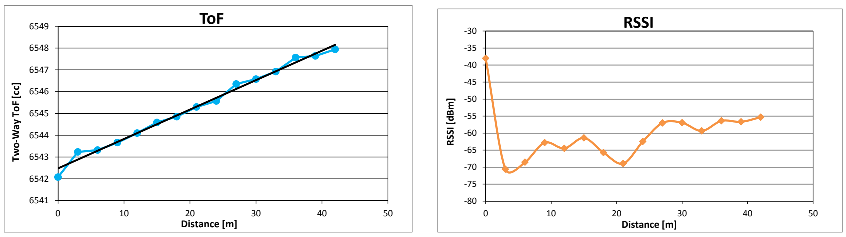


(a) Histogram of 500kbps data rate (b) Histogram of 250kbps data rate

Fig. 7. Timing Histogram of 10000 two-way values

5.3 Comparison between ToF and RSSI

This section provides a comparison between ToF and RSSI, two of the most well established methods for estimating the range between two nodes in RF systems. The CC2500 radio offers the option of capturing the RSSI value of an incoming packet upon reception. That option was used in one of the outdoor experiments and the RSSI value of the reply ToF messages sent by the"responder" to the "requester" was captured. The calculation of the mean RSSI value took place in a similar way like the ToF by averaging 1000 RSSI values. Figure 8 illustrates the ToF and RSSI values against the distance of the two nodes. Typically the RSSI values decay proportionally to d^{-n} where n is the path-loss exponent, normally between and four [18]. From Fig. 8, it is clear that the this relationship is not confirmed and thus the equivalent distance estimation will be faulty. On the other hand the proposed ToF system, demonstrates the expected linearity.



(a) Distance vs ToF (b) Distance vs RSSI

Fig. 8. ToF vs RSSI

Although the approach that is followed for the RSSI does not take into account various factors which cause the attenuation of the signal, like shadowing effects, we provide it here as a comparison to the proposed ToF method.

6 Conclusions

In this paper, we presented a two-way RF-ToF method for ranging estimation in wireless embedded nodes. The multiple two-way transaction approach, achieves two major objectives. Firstly, it does not require the difficult task of synchronization among the participating nodes and secondly, amends the lack of fine resolution due to the low-frequency clocks that most WSNs are equipped with. In our opinion the calibration method we follow is effective since it caters for a number of delays difficult to be measured by using only a clock at the "reply" device. Sub-clock resolution is achieved by averaging the obtained time values. In addition, a simple yet effective procedure disposes any erroneous values that are present in the set of measurements. Experimental results demonstrate an average accuracy of about 1m in outdoors deployments and about 2.25m indoors. Accuracy can be further reduced if additional two-way measurements are used. The proposed ranging system is implemented on COTS hardware. It is therefore our belief that it can be implemented on different hardware platforms. Unlike other ToF ranging methods, our system does not require any additional hardware. The entire procedure of obtaining and filtering the values as well as the calculation of the final ToF is completed on the nodes.

One future direction of this work is to employ the proposed ranging method in scenarios that include mobile nodes. Mobility poses strict latency demands, and this system was designed with this in mind. Although, we have not experimented with tracking of mobile targets yet, we plan this to be one of the application domains of the work presented in this paper.

Acknowledgements. The authors wish to thank Kay Römer for his valuable comments during the review process of this paper and the anonymous reviewers for their constructive feedback.

References

1. Bahl, P., Padmanabhan, V.N.: Radar: an in-building rf-based user location and tracking system. In: IEEE INFOCOM (March 2000)
2. Chung, W.C., Ha, D.: An accurate ultra wideband (uwb) ranging for precision asset location. In: IEEE UWBST (November 2003)
3. Fontana, R., Gunderson, S.: Ultra-wideband precision asset location system. In: IEEE UWBST (May 2002)
4. Gustafsson, F., Gunnarsson, F.: Mobile positioning using wireless networks: possibilities and fundamental limitations based on available wireless network measurements. IEEE Signal Process. Mag. 22(4), 41–53 (2005)

5. Hightower, J., Borriello, G.: Location systems for ubiquitous computing. IEEE Computer 34(8), 57–66 (2001)
6. Karalar, T., Rabaey, J.: An rf tof based ranging implementation for sensor networks. In: IEEE ICC (June 2006)
7. Lanzisera, S., Lin, D., Pister, K.: Rf time of flight ranging for wireless sensor network localization. In: Intl. Workshop on Intelligent Solutions in Embedded Systems (June 2006)
8. Lee, J.Y., Scholtz, R.: Ranging in a dense multipath environment using an uwb radio link. IEEE J. Sel. Areas Commun. 20(9), 1677–1683 (2002)
9. Maróti, M., Kusy, B., Simon, G., Lédeczi, A.: The flooding time synchronization protocol. In: ACM SenSys (November 2004)
10. Maróti, M., Völgyesi, P., Dóra, S., Kusý, B., Nádas, A., Lédeczi, A., Balogh, G., Molnár, K.: Radio interferometric geolocation. In: ACM SenSys (November 2005)
11. Mazomenos, E., Reeve, J., White, N.: Tracking manoeuvring mobile nodes in wireless sensor networks. In: IEEE ICNSC (April 2010)
12. McCrady, D., Doyle, L., Forstrom, H., Dempsey, T., Martorana, M.: Mobile ranging using low-accuracy clocks. IEEE Trans. Microw. Theory Techn. 48(6), 951–958 (2000)
13. Patwari, N., Ash, J., Kyperountas, S., Hero III, A.O., Moses, R., Correal, N.: Locating the nodes: cooperative localization in wireless sensor networks. IEEE Signal Process. Mag. 22(4), 54–69 (2005)
14. Patwari, N., Hero III, A.O., Perkins, M., Correal, N., O'Dea, R.: Relative location estimation in wireless sensor networks. IEEE Trans. Signal Process. 51(8), 2137–2148 (2003)
15. Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The cricket location-support system. In: MobiCom (August 2000)
16. Sallai, J., Balogh, G., Maróti, M., Lédeczi, Á., Kusy, B.: Acoustic ranging in resource-constrained sensor networks. In: ICWN (June 2004)
17. Sivrikaya, F., Yener, B.: Time synchronization in sensor networks: a survey. IEEE Network 18(4), 45–50 (2004)
18. Srinivasan, K., Levis, P.: Rssi is under appreciated. In: EmNets (May 2006)
19. Stüber, G.L.: Principles of Mobile Communication, 2nd edn. Kluwer Academic Publishers, Norwell (2001)
20. Texas Instruments: Ez430-rf2500 development tool user guide (2008), `http://focus.ti.com/lit/ug/slau227e/slau227e.pdf`
21. Texas Instruments: Cc2500 2.4 ghz low-cost low-power transceiver datasheet (2009), `http://focus.ti.com/lit/ds/symlink/cc2500.pdf`
22. Thorbjornsen, B., White, N., Brown, A., Reeve, J.: Radio frequency (rf) time-of-flight ranging for wireless sensor networks. Measurement Science and Technology 21(3), 1–12 (2010)
23. Whitehouse, K., Culler, D.: Calibration as parameter estimation in sensor networks. In: ACM WSNA (September 2002)

Efficient Energy Balancing Aware Multiple Base Station Deployment for WSNs

Sabbir Mahmud, Hui Wu, and Jingling Xue

School of Computer Science and Engineering,
The University of New South Wales, Australia
{sabbirm,huiw,jingling}@cse.unsw.edu.au

Abstract. Energy reduction is one of the major problems in the design of a wireless sensor network (WSN). Multiple base stations can be used to dramatically reduce the energy consumption of sensor nodes. We consider the following problem of deploying k base stations in a wireless sensor network: Given a wireless sensor network where the location of each sensor node is known, partition the whole sensor network into k disjoint clusters and place one base station for each cluster such that the maximum total energy consumption of any cluster is minimised. We propose the first heuristic for this problem. The time complexity of our heuristic is $O(kn^3)$, where n is the number of sensor nodes of the sensor network. In the special case where k is equal to 1, we propose a quadratic-time algorithm for optimally deploying the base station. Our simulation results show that our heuristic is efficient.

1 Introduction

A wireless sensor networks (WSN) consists of a set of sensors nodes that communicate with each other via radio signals. All the sensor nodes works cooperatively to monitor physical or environmental conditions, such as temperature, sound, pressure and motion. The applications of WSNs range from area monitoring, environmental monitoring, to agriculture and structural monitoring. In some applications, such as border surveillance, bushfire detection and traffic control, several thousands of sensor nodes might be deployed over the monitored region. The diameter of the monitored region can be several kilometres.

In wireless sensor networks, sensor nodes are battery powered. Most of the energy of a sensor node is consumed by communications. One key factor for the energy consumption of a sensor node is the communication distance. A sensor node consumes significantly more energy when the communication distance is increased [1]. As a result, multi-hop communication between each sensor and the base station is more desirable in a large scale wireless sensor networks than the single hop communication. In multi-hop communication, a sensor node may spend most of its energy on relaying data packets. Hence, it is important to shorten the hop distance between each source sensor node and the base station. The hop distances can be greatly reduced by deploying multiple base stations. All the sensor nodes are partitioned into multiple disjoint clusters with one base station for each cluster. Each sensor node sends its data only to its designated base station. Moreover, the location of the base station of each cluster is very important. If the base station is deployed far from the data sources, many sensor nodes are required to relay data packets and the energy consumption of those sensor nodes will be significantly increased. Therefore, it is an important design issue to find the best location of a base station. Nevertheless, the problem of optimally deploying multiple base stations can be reduced to the k-center problem which is NP-complete [12]. Therefore, a polynomial time algorithm is unlikely to exist.

In this paper, we study the problem of deploying k base stations such that the total energy consumption of a WSN is uniformly distributed among all the clusters. Under our energy consumption model, the total energy consumption of each cluster is a monotonically increasing function of the total shortest hop distance of all the sensor nodes of the cluster. The longer the total shortest hop distance, the more the energy consumption of a cluster. In the special case where there is only one base station, we propose a quadratic-time algorithm to optimally place a base station such that the total energy consumption of all the sensor nodes is minimised. Based on the optimal algorithm for the single base station problem, we propose a novel heuristic that aims to partition all the sensor nodes into k disjoint clusters such that the maximum total energy consumption of any cluster is minimised. We have simulated our heuristic on 195 instances of different distributions. Our simulation results show that our heuristic is very effective.

2 Definitions and Network Model

A wireless sensor network consists of a set of n identical sensor nodes each of which is located in a $2D$ plane. The location of each sensor node is known. All the sensor nodes have the same maximum communication distance R. We assume that there are no communication barriers between any two adjacent sensor nodes[1]. Therefore, a sensor node v_i can directly communicate with a sensor node v_j if the Euclidean distance between v_i and v_j is not greater than R. There are k base stations to be deployed in a target WSN. As a result, all the sensor nodes need to be partitioned into k clusters with one base station for each cluster. A sensor node in each cluster sends its data to its base station only. If the Euclidean distance between a sensor node and its base station is greater than R, the data of the sensor node must be transmitted via other sensor nodes to the base station.

Definition 1. *The connectivity graph of a wireless sensor network is a undirected graph $G =< V, E >$, where $V = \{v_i : i = 1..n \text{ and } v_i \text{ is a sensor node}\}$, and $E = \{(v_i, v_j) : \text{if the Euclidean distance between } v_i \text{ and } v_j \text{ is not greater than } R\}$.*

[1] Our approach can be modified to handle the communication barriers.

Without loss of generality, we assume that the connectivity graph G of the target wireless sensor network is connected.

Definition 2. *Given two sensor nodes v_i and v_j, the shortest hop distance from v_i to v_j is the length of the shortest path from v_i to v_j in the connectivity graph.*

The shortest hop distance of a sensor node v_i to the base station gives the lower bound on the number of hops of a packet transmitted from v_i to the base station.

Definition 3. *Given a cluster of sensor nodes and a base station, the total shortest hop distance of the cluster is the sum of all the shortest hop distances from each sensor node to the base station.*

Let P be a set of n distinct points called sites, in a $2D$ plane. The Voronoi diagram [11] of P is the subdivision of the plane into n cells, one for each site. A point q lies in the cell of a site $p_i \in P$ iff the Euclidean distance between q and p_i is less than the Euclidean distance between q and p_j ($p_j \in P$ and $i \neq j$). The edges of the Voronoi diagram are all the points in the plane that are equidistant to the two nearest sites.

Definition 4. *A sensor node v_i is a neighbour of a sensor node v_j if the Voronoi cells of v_i and v_j share a Voronoi edge.*

Definition 5. *Let V be a set of n sensor nodes in a $2D$ plane and $C_i(i = 1, 2, \cdots, k)$ be k disjoint clusters of V. A cluster C_i is a neighbour of a cluster C_j if there are two sensor nodes $v_s \in C_i$ and $v_t \in C_j$ such that v_s is a neighbour of v_t.*

Definition 6. *Given a cluster C_i of sensor nodes and a sensor node $v_j \notin C_i$, the Euclidean distance from v_j to C_i, denoted $d(v_j, C_i)$, is $\min\{d(v_k, v_j) : v_k \in C_i$ and $d(v_k, v_j)$ is the Euclidean distance between v_k and $v_j\}$.*

Definition 7. *Given a wireless sensor network and a point p on a $2D$ plane, the unit sensor density of p is the number of sensor nodes that are one hop away from p. The maximum unit sensor density of the wireless sensor network is the largest unit sensor density of all the points on the $2D$ plane.*

Throughout this paper, we assume that the maximum unit sensor density is a constant. In wireless sensor networks, the maximum communication distance is typically short in order to reduce the energy consumption of data transmissions. Hence this assumption is reasonable.

3 An Optimal Algorithm for Single Base Station Deployment Problem

Deploying a single base station in a cluster is a building block of our heuristic for optimally deploying k base stations. This problem is described as follows. Given a cluster of sensor nodes and a base station, find the optimal location of the base station such that the total shortest hop distance of the cluster is minimised. Next, we will propose an efficient algorithm for this problem.

The key idea of our algorithm is to find the candidate locations of the base station such that one candidate location must be the optimal location of the base station. To find all possible candidate locations, we consider each pair of sensor nodes v_i and v_j. If the Euclidean distance between v_i and v_j is greater than $2R$, where R is the maximum communication distance of all the sensor nodes, we will ignore the pair v_i and v_j. Otherwise, we find the candidate circles of v_i and v_j. A candidate circle of v_i and v_j is a circle that satisfies the following two constraints:

1. The radius of the circle is R.
2. v_i and v_j are on its circumference.

The centre of a candidate circle is a candidate location of the base station. Notice that for each pair of sensor nodes at most two candidate circles exist. If the Euclidean distance of a pair of sensor nodes is equal to $2R$, only one candidate circle of this pair exists. After finding all the candidate locations, our algorithm will search for the best candidate location of the base station. The best candidate location is the one that minimises the total shortest hop distance of all the sensor nodes to the base station placed at this candidate location. The algorithm is shown as follows.

```
Algorithm OptimalD($V$)
Input : A set $V = \{v_1, v_2, \cdots, v_m\}$ of $m$ sensor nodes in a 2D plane and a base
station.
Output : The optimal location of the base station such that the total shortest
hop distance of all the sensor nodes to the base station at the optimal location
is minimised, and the resulting total shortest hop distance.
begin
  $C = \emptyset$;
  for each pair of sensor nodes $(v_i, v_j)(v_i, v_j \in V)$ do
      if the Euclidean distance between $v_i$ and $v_j \leq 2R$ then
          Find the candidate circles $C_1$ and $C_2$ of $v_i$ and $v_j$;
          Let $c_1$ and $c_2$ be the centres of $C_1$ and $C_2$;
          $C = C \cup \{c_1\} \cup \{c_2\}$;
  for each candidate location $c_i \in C$ do
      Place the base station at $c_i$;
      Construct the connectivity graph $G(V \cup \{c_i\})$ of all the sensor nodes
      and the base station;
      Compute the total shortest hop distance TSHD($c_i$) of all the sensor nodes in $V$
      to the base station located at $c_i$;
  Let $c_j$ be the candidate location with the minimum total shortest hop distance;
  return ($c_j$,TSHD($c_j$));
end
```

Theorem 1. *Given a cluster of m sensor nodes, the time complexity of the algorithm OptimalD(V) is $O(m^2)$.*

Theorem 2. *The algorithm OptimalD(V) is guaranteed to find the optimal location of the base station.*

Proof. Assume that the optimal location is c_{opt}. Let $S = \{v_1, v_2, \cdots, v_r\}$ be the set of sensor nodes that are one hop away from the base station at the optimal location c_{opt}. Draw a circle C_{opt} with the radius R and the centre c_{opt}. According to the definition of the maximum communication distance R, all the sensor nodes in S must be either in C_{opt} or on the circumference of C_{opt}. Next, we show that there is a candidate location c_k generated by our algorithm such that the set of sensor nodes that are one hop away from c_k is equal to S. Consider the following three possible cases.

1. There are two sensor nodes $v_i, v_j \in S$ such that v_i and v_j are on the circumference of C_{opt}. In this case, c_{opt} is one of our candidate locations.
2. Only one sensor node $v_i \in S$ is on the circumference of C_{opt}. Turn the circle C_{opt} clockwise around v_i until another sensor $v_j \in S$ is on the circumference of C_{opt}. Now all the sensor nodes in S are still in C_{opt} and this case reduces to Case 1.
3. No sensor node is on the circumference of C_{opt}. Arbitrarily select a sensor node v_t, and move C_{opt} along the straight line $c_{opt}v_t$ until one sensor node in S is on the circumference of C_{opt}. Now all the sensors in S are still in C_{opt} or on the circumference of C_{opt}. Hence, this case reduces to Case 2.

Based on the above discussions, we can conclude that such a candidate location c_k exists. For each sensor node v_i, any path from v_i to c_k or c_{opt} must include a sensor node in S. Therefore, the shortest hop distance from v_i to c_k is equal to that from v_i to c_{opt}. As a result, c_k is also an optimal location of the base station.

4 Incremental Algorithms for Single Base Station Deployment Problems

Our heuristic for k base station deployment problem needs to repeatedly find the optimal location of a base station for a growing or shrinking cluster. A growing cluster is a cluster of sensor nodes such that a new sensor node is added to it at a time. A shrinking cluster is a cluster of sensor nodes such that a sensor node is removed from it at a time. There are two single base station deployment problems: the single base station deployment problem for a growing cluster and the single base station deployment problem for a shrinking cluster.

The single base station deployment problem for a growing cluster is described as follows: Given a cluster C_i of sensor nodes, a new sensor node v_k, and a base station, find the optimal location of the base station such that the total shortest hop distance from all sensor nodes in $C_i \cup \{v_k\}$ to the base station is minimised. A bruteforce approach to this problem is to use the algorithm proposed in the previous section, which takes $O(m^2)$ time, where m is the number of sensor nodes of the cluster. Next, we propose a faster incremental algorithm which takes $O(m)$ time.

Let $B(C_i)$ be the set of all candidate locations of the base station for C_i, SHD(v_i, v_j) the shortest hop distance between v_i and v_j, and $N(c_j)$ the set of all neighbouring sensor nodes which are one hop away from a candidate location c_j. The incremental algorithm for the single base station deployment problem for a growing cluster is shown as follows.

```
Algorithm IncrementalGrowing($C_i, v_k$)
Input : A cluster $C_i$, the set $B(C_i)$ of all candidate locations of the base station for
$C_i$, the total shortest hop distance TSHD$(c_j)$ of each candidate location $c_j$ of $C_i$,
the neighbour set $N(c_j)$ of each candidate location $c_j$, and a new node $v_k$.
Output : The optimal location of the base station for $C_i \cup \{v_k\}$, the set $B(C_i)$ of all
candidate locations of the base station for $C_i \cup \{v_k\}$, the total shortest hop distance
TSHD$(c_j)$ of each candidate location $c_j$ of $C_i \cup \{v_k\}$, and the neighbour set $N(c_j)$
of each candidate location $c_j$ of $C_i \cup \{v_k\}$.
begin
  for each neighbouring candidate location $c_j$ of $v_k$ do
      $N(c_j) = N(c_j) \cup \{v_k\}$;
  Find the shortest hop distance SHD$(v_k, v_j)$ from $v_k$ to each sensor node $v_j \in C_i$;
  Construct the set $A$ of all the new candidate locations generated by $v_k$ and its
  neighbouring sensor nodes;
  for each new candidate location $c_j \in A$ do
      Find $N(c_j)$;
      Find the total shortest hop distance TSHD$(c_j)$ from all sensor nodes
      in $C_i \cup \{v_k\}$ to $c_j$;
  for each candidate location $c_j \in B(C_i)$ do
  // Compute the total shortest hop distance of each candidate location.
      SHD$(c_j, v_k) = 1 + \min\{SHD(v_s, v_k) : v_s \in N(c_j)\}$;
      TSHD$(c_j)$ =TSHD$(c_j)$+SHD$(c_j, v_k)$;
  $B(C_i) = B(C_i) \cup A$;
  Find the optimal location $c_o$ of the base station with the smallest total shortest
  hop distance;
  return ($c_o$,TSHD$(c_o)$);
end
```

Theorem 3. *The time complexity of IncrementalGrowing(C_i, v_k) is $O(m)$.*

The single base station deployment problem for a shrinking cluster is described as follows: Given a cluster C_i of sensor nodes, a sensor node $v_k \in C_i$, and a base station, find the optimal location of the base station for the cluster $C_i - \{v_k\}$ such that the total shortest hop distance from all sensor nodes in $C_i - \{v_k\}$ to the base station is minimised. A fast incremental algorithm is shown as follows.

```
Algorithm IncrementalShrinking($C_i, v_k$)
Input : A cluster $C_i$, the set $B(C_i)$ of all candidate locations of the base station for
$C_i$, the total shortest hop distance TSHD$(c_j)$ of each candidate location $c_j$, $N(c_j)$,
and a node $v_k \in C_i$.
Output : The optimal location of the base station for $C_i - \{v_k\}$, the set $B(C_i)$ of all
candidate locations of the base station for $C_i - \{v_k\}$, the total shortest hop distance
TSHD$(c_j)$ and the neighbour set $N(c_j)$ of each candidate location $c_j$ of $C_i - \{v_k\}$.
begin
  Find the shortest hop distance SHD$(v_k, v_j)$ from $v_k$ to each sensor node $v_j \in C_i$;
  Compute the set $A$ of all the candidate locations which are solely generated by $v_k$
  and its neighbouring sensor nodes;
  $B(C_i) = B(C_i) - A$;
  **for** each neighbouring candidate $c_j$ that is one hop away from $v_k$ **do**
      $N(c_j) = N(c_j) - \{v_k\}$;
  $C_i = C_i - \{v_k\}$;
  **for** each candidate location $c_j \in B(C_i)$ **do** ;
      // Compute the total shortest hop distance of each candidate location.
      $SHD(c_j, v_k) = 1 + \min\{SHD(v_s, v_k) : v_s \in N(c_j)\}$;
      $TSHD(c_j) = TSHD(c_j) - SHD(c_j, v_k)$;
  find the optimal location $c_o$ of the base station with the minimum shortest
  hop distance;
  **return** $(c_o, TSHD(c_o))$;
**end**
```

Theorem 4. *The time complexity of IncrementalShrinking(C_i, v_k) is $O(m)$.*

5 A Heuristic for the Optimal *k* Base Station Deployment Problem

Given k base stations and a set of sensor nodes in the $2D$ plane, the energy balancing aware k base station deployment problem is to partition the whole sensor network into k disjoint clusters and deploy a base station for each cluster in an optimal way such that the maximum total shortest hop distance of any cluster is minimised. Similar to the k-center problem [12], this problem is NP-complete. Next, we will propose an efficient heuristic for this problem.

Conceptually, our heuristic works in two phases. In the first phase, it creates k initial disjoint clusters by using a greedy approach. In the second phase, it keeps moving a sensor node from a cluster with a larger total shortest hop distance to a neighbouring cluster with the smaller total shortest hop distance until a fixed point is reached.

Next, we describe how each phase works in details. In the first phase, the algorithm CreatingClusters(V, k) creates k initial disjoint clusters. It starts with creating the Voronoi diagram of all the sensor nodes. The Voronoi diagram is used to determine the nearest sensor node of a cluster. Initially, there are n clusters with one sensor node in each cluster, and the total shortest hop distance of each cluster is 0. Next, it repeatedly finds a cluster with the smallest total shortest hop distance and merges it with the best neighbouring cluster until only k clusters are left. The best neighbouring cluster is the neighbouring cluster that minimises the total shortest hop distance of the resulting cluster merged from these two clusters. The pseudo code of the algorithm is shown as follows.

```
**Algorithm** CreatingClusters$(V, k)$
**Input** : A set $V = \{v_1, v_2, \cdots, v_n\}$ of sensor nodes and $k$ base stations.
**Output** : The $k$ disjoint clusters and the optimal location of the base station of each
cluster.
**begin**
  Create the Voronoi diagram for all sensor nodes in $V$;
  **for** each $v_i \in V$ **do**
```

```
        $C_i = \{v_i\}$; $TSHD(C_i) = 0$;
    $NumberOfCluster = n$;
    while $NumberOfCluster > k$ do
        Select a cluster $C_i$ with the minimum $TSHD(C_i)$.
        Fnd all the neighbouring clusters of the cluster $C_i$;
        for each neighbouring cluster $C_j$ of $C_i$ do
            $tempC_{ij} = C_i \cup C_j$;
            ($c_{ij}$, $TSHD(tempC_{ij}) = OptimalD(tempC_{ij})$;
        Find the neighbouring cluster $C_j$ that has the minimum $TSHD(tempC_{ij})$;
        Merge $C_i$ and $C_j$ into a new cluster $C_{ij}$;
        $TSHD(C_{ij}) = TSHD(tempC_{ij})$;
        $NumberOfCluster = NumberOfCluster - 1$;
end
```

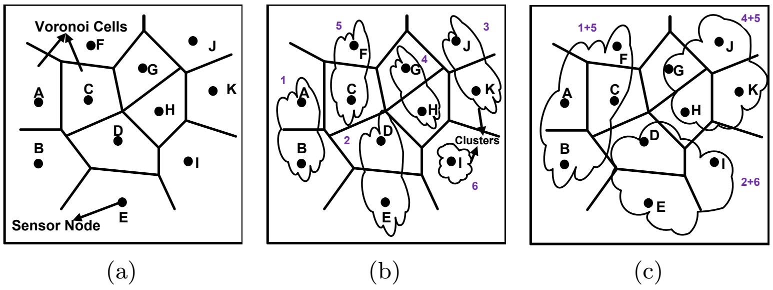


Fig. 1. An example for illustrating the algorithm CreatingClusters(V, k)

We use an example to illustrate how the algorithm CreatingClusters(V, k) works. Consider a wireless sensor network with 11 sensor nodes and 3 base stations as shown in Fig. 1. Fig. 1(a) shows the Voronoi diagram our algorithm creates. All the neighbouring nodes of I are E, D, H, and K, and all the neighbouring nodes of H are C, D, G, I, J and K. At the beginning, there are 6 clusters with each sensor being one cluster. Next, the algorithm merges a smallest cluster with its best neighbouring cluster at a time. Figure 1(b) shows the intermediate clusters created by the algorithm CreatingClusters(V, k), where each cluster except the cluster I is merged from two clusters. For example, the cluster $\{C, F\}$ is merged from the cluster $\{C\}$ and the cluster $\{F\}$. Figure 1(c) shows the final clusters $\{A, B, C, F\}$, $\{D, E, I\}$ and $\{G, J, H, K\}$ created by our algorithm.

In the second phase, the algorithm ClusterBalancing(C, L) aims to modify the initial clusters so that the maximum total shortest hop distance of any cluster is minimised, where C is the set of k initial clusters and L is the set of the optimal locations of the k base stations. It starts with the initial k clusters created by the algorithm CreatingClusters(V, k). In each iteration, a modifiable cluster C_i with the highest TSHD among all the clusters in C is selected. A cluster C_i is modifiable if there exist a neighbouring cluster

C_s with TSHD(C_s) <TSHD(C_i) and a sensor node $v_k \in C_i$ such that TSHD($C_s \cup \{v_k\}$) <TSHD(C_i) and TSHD($C_i - \{v_k\}$) <TSHD(C_i) hold, i.e., moving v_k from C_j to C_s will reduce the maximum total shortest hop distance of both clusters. If such a modifiable cluster does not exist, all the clusters are balanced and the algorithm terminates. If such a modifiable cluster C_i exists, the algorithm will select the neighbouring cluster C_j with the smallest TSHD among all the neighbouring clusters of C_i and find the set Q of sensor nodes in C_i which are the neighbouring sensor nodes of C_j. Then it keeps moving a sensor node in Q with the smallest Euclidean distance to C_j from C_i to C_j until no sensor node in Q can be moved from C_i to C_j. A sensor node $v_k \in Q$ is moved from C_i to C_j only if v_k satisfies the following constraints:

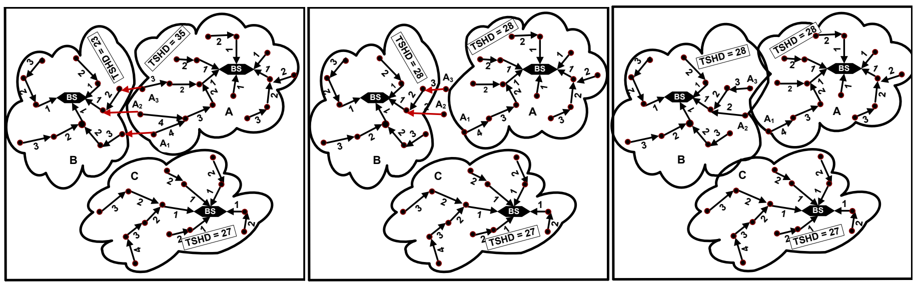
1. TSHD($C_i - \{v_k\}$)<TSHD(C_i).
2. TSHD($C_j \cup \{v_k\}$)<TSHD(C_i).

The first constraint ensures that after v_k is moved from C_i to C_j, the total shortest hop distance of C_i is reduced. If a sensor node $v_s \in Q$ is on the shortest paths of other sensor nodes in C_i to the base station, moving v_s from C_i to C_j may increase the total shortest hop distance of C_i. The second constraint guarantees that after moving v_k from C_i to C_j, the total shortest hop distance of C_j will be less than the previous total shortest hop distance of C_i. The algorithm is shown in pseudo code as follows.

```
Algorithm ClusterBalancing($C, L$)
Input : A set $C = \{C_1, \cdots, C_k\}$ of $k$ disjoint clusters and a set $L = \{c_1, \cdots, c_k\}$ of
the optimal locations of $k$ base stations, where $c_i (i = 1, 2, \cdots, k)$ is the optimal
location of the base station of the cluster $C_i$.
Output : A new set of $k$ disjoint clusters with smaller maximum total shortest hop
distance and the optimal location of the base station of each cluster.
begin
  // $A$ is the set of clusters from which no sensor node can be moved.
  // $B$ is the set of clusters from which a sensor node might be moved.
  $A = \{\}$; $B = C$; // Note that $C = A \cup B$ holds all the time.
  for each cluster $C_i \in C$ do
      $modifiable(C_i) = true$;
  while $B \neq \emptyset$ do
      Select a cluster $C_i$ with the maximum TSHD($C_i$)
      and $modifiable(C_i) = true$ from $B$;
      $S = \{C_s : C_s \in C$ and $C_s$ is a neighbouring cluster of $C_i\}$.
      Find $C_j \in S$ with the minimum TSHD($C_j$);
      $Q = \{v_s : v_s \in C_i$ and $v_s$ is a neighbouring sensor node of $C_j\}$;
      NodeMoved($C_i$) = 0;
      while $Q \neq \emptyset$ do
          Select a sensor node $v_s \in Q$ with the smallest Euclidean distance to $C_j$;
          if TSHD($C_i - \{v_s\}$)<TSHD($C_i$) && TSHD($C_j \cup \{v_s\}$)<TSHD($C_i$) then
              $C_j = C_j \cup \{v_s\}$; $C_i = C_i - \{v_s\}$;
              Find the new optimal locations of the base stations of $C_i$ and $C_j$;
              Recalculate TSHD($C_i$) and TSHD($C_j$);
              NodeMoved($C_i$) = 1;
          $Q = Q - \{v_s\}$;
```

```
    **if** $NodeMoved(C_i) > 0$ **then**
        **for** *each neighbouring cluster* $C_j$ *of* $C_i$ **do**
            **if** $modifiable(C_j) == false$ **then**
                $modifiable(C_j) = true;\ A = A - \{C_j\};\ B = B \cup \{C_j\};$
    **else**
        $modifiable(C_i) = false;\ A = A \cup \{C_i\};\ B = B - \{C_i\};$
**end**
```

Now we use an example to illustrate how the algorithm ClusterBalancing(C, L) works. In Fig. 2, there are three clusters A, B and C. The cluster A has the largest total shortest hop distance which is 35. A has two neighbouring clusters: clusters B and C. The total shortest hop distance of B is less than that of C. Therefore, the neighbouring sensor nodes from A will be moved to B. In Fig. 2(a),



(a) (b) (c)

Fig. 2. An example for illustrating the algorithm ClusterBalancing(C, L)

the neighbouring sensor nodes in the cluster A are A_1, A_2, and A_3. A_3 is closest to the cluster B. TSHD($A - \{A_3\}$) is 31 which is smaller than TSHD(A). So it satisfies the first constraint. TSHD($B \cup A_3$) = $(23 + 3) = 26$, is smaller than TSHD(A). So, A_3 satisfies the second constraint. As a result, A_3 is moved to the cluster B. Subsequently, the sensor node A_2 satisfies both constraints and it is also moved to the cluster B as shown in Fig. 2(b). However, the sensor node A_1 does not satisfy the first condition. So it is not moved to the cluster B.

Next we analyse the time complexities of the two algorithms used by our heuristic.

Theorem 5. *The time complexity of CreatingClusters*(V, k) *is* $O(n^2 \log n)$.

Proof. Given n sensor nodes, its Voronoi diagram can be constructed in $O(n \log n)$ time [11]. The number of neighbouring cluster of each cluster is at most p, where p is the maximum unit sensor density. Under our assumption, p is a constant. Therefore, it takes $O(1)$ time to find all the neighbouring clusters of each cluster. Each merge takes $O(s^2)$ time if the number of sensor nodes of the resulting cluster is s. The whole merge process of clusters can be represented by a merge tree, where each node denotes merging two clusters into one cluster. At each level of the merge tree, the total work is $O(n^2)$. Since the merge tree is a balanced tree, its height is at most $\log n$. Therefore, the total work of the whole merge process is $O(n^2 \log n)$. As a result, the time complexity of the algorithm CreatingClusters(V, k) is $O(n^2 \log n)$.

Theorem 6. *The time complexity of ClusterBalancing(C, L) is $O(kn^3)$, where n is the number of sensor nodes of the wireless sensor network.*

Proof. It takes at most $k-1$ sensor motions to reduce the number of sensor nodes of the cluster with the maximum total shortest hop distance by one. Therefore, the total number of sensor node motions is bounded by $O(kn)$. For each sensor motion, it takes $O(n^2)$ time to find the sensor node v_s of C_i that has the shortest Euclidean distance to C_j by using exhaustive search, and $O(n)$ time to move v_s from C_i to its neighbouring cluster C_j by using our incremental algorithms IncrementalGrowing(C_i, v_k) and IncrementalShrinking(C_i, v_k). Therefore, the time complexity of ClusterBalancing$(C,\ L)$ is $O(kn^3)$.

6 Related Work

Deploying multiple base stations in a large scale senor network can significantly decrease the energy consumption of the sensor nodes by shortening the distance between the source sensor nodes and the base station. The problems of finding the best locations of multiple base stations have been studied in a number of papers under different optimisation objectives. Most papers formulate the problems as an integer linear programming (ILP) problem [6,2,3]. [6] proposes a heuristic for deploying multiple mobile base stations to maximise the lifetime of the sensor network. The total lifetime of the network is divided into equal period of time known as rounds and all mobile base stations change their locations at the beginning of every round. An ILP formulation is proposed to find the locations of base stations such that the maximum energy spent by each node in a round is minimised. [2] proposes a heuristic for maximising the life time of a WSN. The heuristic consists of a LP formulation for positioning multiple base stations in a sensor network and an ILP formulation for routing traffic flow from all of the sensors to these multiple sink nodes. Since the ILP problem is NP-complete, the ILP-based approaches are not applicable to large scale WSNs.

[3] proposes two-tier WSNs where the entire network is divided into clusters and each cluster has its own cluster head which is responsible for transferring data to the base station after collecting data from the sensor nodes. An iterative algorithm SPINDS is proposed to iteratively move the cluster head to a better location in order to increase the life time of the WSN. [7] studies the problems of hybrid sensor networks with resource-rich (micro-servers) and resource-deprived sensor nodes. An iterative tabu-search based algorithm is proposed to find the best locations of the micro-servers.

[10] propose an algorithm and a heuristic for placing k base stations in an optimal way such that the average Euclidean distance between the sensor nodes and their base stations is minimised. The algorithm assumes that each base station

knows the locations of all sensor nodes and the heuristic assumes that each base station only knows the locations of its neighbouring sensor nodes and other base stations. In WSNs, it is possible for a sensor node with a shorter Euclidean distance to its base station to have a longer hop distance to its base station. Even worse, it is possible that no sensor node can communicate with the base station at the location that minimises the average Euclidean distance of all the sensor nodes to the base station. Consider a WSN with a ring topology, i.e., all the senor nodes are located on a ring. If the radius of the ring is greater than the maximum communication distance of the sensor nodes, no sensor nodes can communicate with the base station at the center of the ring. As a result, it is not feasible to minimise the average Euclidean distance between the sensor nodes and their base stations in order to minimise the lifetime or the total energy consumption of a WSN.

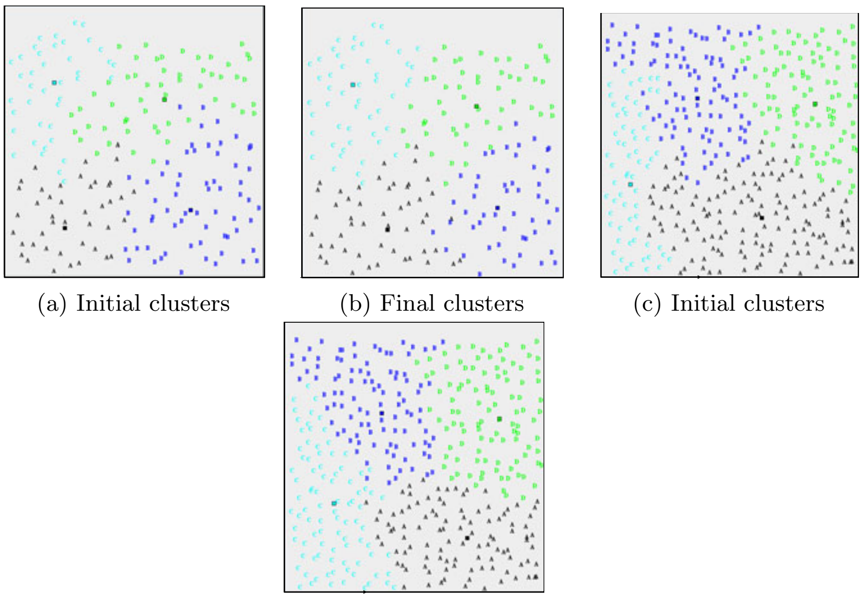
[9] studies the problem of placing k base stations in an optimal way such that the total latency of all the sensor nodes to their gateways is minimised. The authors proposed two heuristics for the problem using genetic algorithms. The problem with minimising the total latency of all the sensor nodes to their gateways is that it may result in unbalanced energy consumption of all the clusters.

In a WSN with multiple base stations, base stations should be deployed such that the total energy consumption of the whole WSN is distributed uniformly among all the clusters in order to increase the life time of the WSN. To our knowledge, no previous research on deploying multiple base stations with such an optimisation objective has been reported. Our work presented in this paper is the first one on uniformly distributing the total energy consumption among all the clusters.

7 Simulation Results

In this section, we evaluate the performance of our heuristic via simulations. We have generated 195 different network instances. The WSN represented by each instances is connected. All these instances are classified into three categories: grid, uniform distribution and random distribution. We have used three different numbers of base stations, i.e., 2, 4 and 6. For uniform and random distributions, we have varied the number of sensor nodes from 100 to 600 with an increment of 20 sensor nodes. For either distribution, we have generated 25 instances with 3 different numbers of base stations. For grid, we have generated 45 instances.

In order to simulate our heuristic, we have used a computer with Intel Core 2 Duo processor. The processor has a clock frequency of 3 GHz and 4 GB RAM. In order to measure the performance of our heuristic, we have introduced a metric named *unbalance factor*. Given a WSN N with k clusters, the unbalance factor of N is defined as $(\text{TSHD}_{max} - \text{TSHD}_{min}) / \text{TSHD}_{max}$, where $\text{TSHD}_{max} = \max\{\text{TSHD}(C_i) : C_i \in N\}$ and $\text{TSHD}_{min} = \min\{\text{TSHD}(C_i) : C_i \in N\}$. The unbalance factor shows how unbalanced the k clusters of a WSN are. The smaller



(a) Initial clusters (b) Final clusters (c) Initial clusters

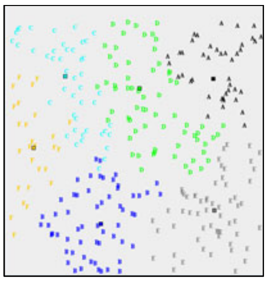
(d) Final clusters

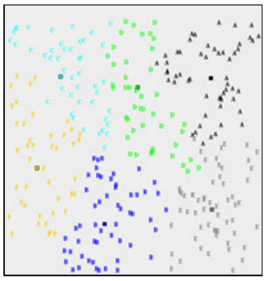
Fig. 3. Simulation results for 200 and 400 sensor nodes with 4 base stations in uniform distribution

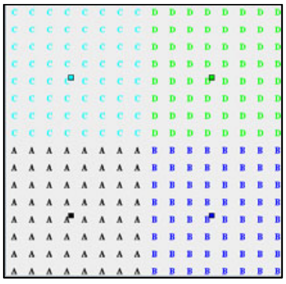
the unbalance factor, the more balanced the k clusters. If the unbalance factor is 0, the k clusters are fully balanced. We have also recorded the running time of our heuristic for each instance we have generated.

In order to give readers some intuition on the performance of our heuristic, we have randomly selected 4 instances among all 195 instances we have generated. The simulations results of these 4 instances are shown in Figs. 3 and 4, where the sensor nodes in the same cluster are shown in the same letter and colour, and a square denotes a base station. Figure 3(a) shows the initial clusters of 200 sensor nodes with 4 base stations, generated by our algorithm CreatingClusters(V, k). Figure 3(b) shows the final clusters generated by our algorithm ClusterBalancing(C, L), where all the clusters are almost balanced. Figures 3(c) and 3(d) show the initial clusters and the final clusters, respectively, of 400 uniformly distributed sensor nodes with 4 base stations. Figures 4(a) and 4(b) shows the simulation results for 300 sensor nodes with 6 base stations where sensor nodes are deployed in random distribution. Figures 4(c) and 4(d) show the initial clusters and the final clusters of 256 sensor nodes and 4 base stations deployed in grid, where both clusters are fully balanced.

The complete simulation results are shown in Figs. 5 and 6. Figure 5 shows the unbalance factors of the clusters constructed by our heuristic for all the instances we have generated. Overall, all the clusters constructed by our heuristic are well balanced. We can observe the following patterns:


(a) Initial clusters


(b) Final clusters


(c) Initial clusters

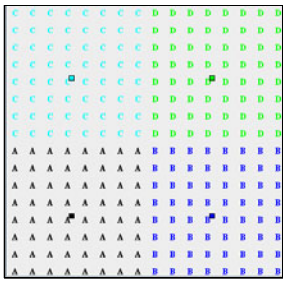
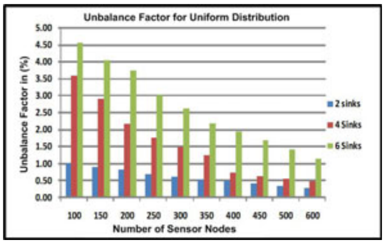
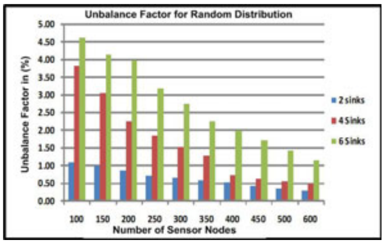

(d) Final clusters

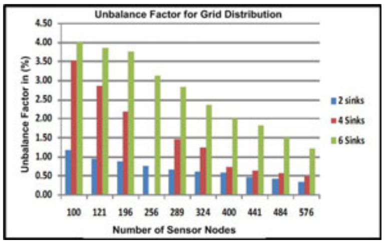
Fig. 4. Simulation results for 300 and 256 sensor nodes with 6 and 4 base stations in random and grid distributions



(a)



(b)



(c)

Fig. 5. Unbalance Factor for sensor nodes in uniform, random and grid distributions

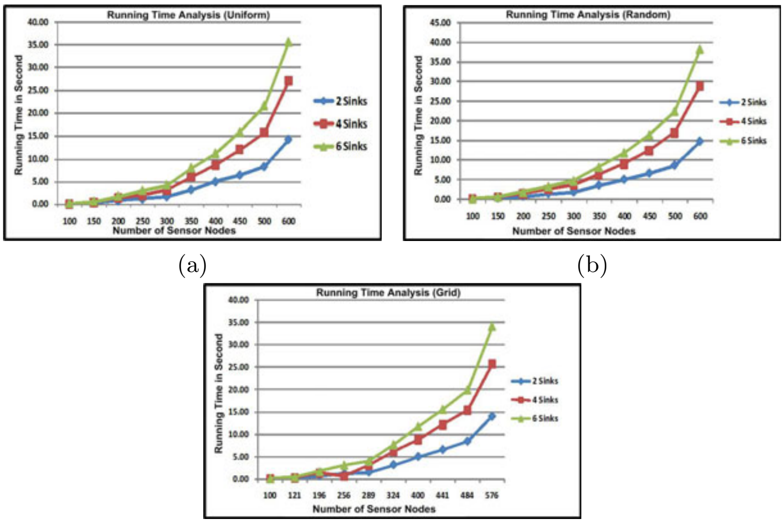


Fig. 6. Running times for all the instances

1. The unbalance factor increases with the number of base stations. The unbalance factor is at most 1% when the number of base stations is 2. The reason is that when the number of base stations increases, the relative difference between the cluster with the longest shortest hop distance and the cluster with the smallest shortest hop distance will increase.
2. Given a fixed number of base stations, the unbalance factor decreases with the number of sensor nodes. This is because the maximum total shortest hop distance of any cluster increases with the number of sensor nodes.

Figure 6 shows the running times in second of our heuristic for different instances. It shows that running time increases approximately cubically with the number of sensor nodes in all the three distributions, which is consistent with the time complexity of our heuristic. For the instance that has 600 sensor nodes in random distribution and 6 base stations, our heuristic took around 38 seconds, which is the longest running time.

8 Conclusion

In this paper, we proposed the first heuristic for optimally deploying k base stations in a WSN such that the maximum total shortest hop distance of any cluster is minimised. The time complexity of our heuristic is $O(kn^3)$, where n is the number of sensor nodes of the WSN. In the spacial case where there is only one base station, we proposed an optimal algorithm for this problem. We have performed simulations of our heuristic on 195 instances. The simulation results show that our heuristic is very effective.

Although our heuristic performs very well, its approximation ratio is unknown. We conjecture that it is at most 2. In the future work, we will find the approximation ratio of our heuristic. Another open problem is to optimally deploy multiple base stations in a WSN where sensor nodes have variable communication ranges.

References

1. Gao, Q., Blowa, K.J., Holdinga, D.J., Marshallb, I.W., Penga, X.H.: Radio range adjustment for energy efficient wireless sensor networks. Ad Hoc Networks 4(1), 75–82 (2006)
2. Kim, H., Seok, Y., Choi, N., Choi, Y., Kwon, T.: Optimal multisink positioning and energy-efficient routing in Wireless Sensor Networks. In: Kim, C. (ed.) ICOIN 2005. LNCS, vol. 3391, pp. 264–274. Springer, Heidelberg (2005)
3. Hou, Y.T., Shi, Y., Sherali, H.D., Midkiff, S.F.: On energy provisioning and relay node placement for wireless sensor networks. IEEE Transactions on Wireless Communications 4(5), 2579–2590 (2005)
4. Liu, J., Reich, J., Zhao, F.: Collaborative in-network processing for target tracking. EURASIP Journal on Applied Signal Processing, 378–391 (2003)
5. Efrat, A., Har-Peled, S., Mitchell, J.S.B.: Approximation Algorithms for Two Optimal Location Problems in Sensor Networks. In: Proceedings of the 3rd International Conference on Broadband Communications, Networks and Systems (2005)
6. Gandham, S.R., Dawande, M., Prakash, R., Venkatesan, S.: Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In: Proceedings of IEEE Global Telecommunications Conference, vol. 1, pp. 377–381 (2003)
7. Hu, W., Chou, C.T., Jha, S., Bulusu, N.: Deploying long-lived and cost effective hybrid sensor networks. In: 1st Workshop on Broadband Advanced Sensor networks
8. Qiu, L., Chandra, R., Jain, K., Mahdian, M.: Optimizing the placement of integration points in multi-hop wireless sensor networks. In: Proceedings of International Conference on Network Protocols, ICNP (2004)
9. Yousef, W., Younis, M.: Intelligent gateways placement for reduced data Latency in Wireless Sensor Networks. In: IEEE International Conference on Communications (ICC), pp. 3805–3810 (2007)
10. Vincze, Z., Vida, R., Vidacs, A.: Deploying multiple sinks in multi-hop Wireless Sensor Networks. In: IEEE International Conference on Pervasive Services (ICPS), pp. 55–63 (2007)
11. Sack, J.R., Urrutia, J.: Handbook of Computational Geometry. Elsevier Science, Netherlands (2000)
12. Hochbaum, D.S., Shmoys, F.B.: A best possible heuristic for the k-center problem. Mathematics of Operations Research 16(2), 180–184 (1985)

BurstProbe: Debugging Time-Critical Data Delivery in Wireless Sensor Networks

James Brown[1], Ben McCarthy[1], Utz Roedig[1], Thiemo Voigt[2], and Cormac J. Sreenan[3]

[1] Lancaster University, UK
{j.brown,b.mccarthy,u.roedig}@lancaster.ac.uk
[2] Swedish Institute of Computer Science (SICS), Sweden
thiemo@sics.se
[3] University College Cork, Ireland
cjs@cs.ucc.ie

Abstract. In this paper we present *BurstProbe*, a new technique to accurately measure link burstiness in a wireless sensor network employed for time-critical data delivery. Measurement relies on shared probing slots that are embedded in the transmission schedule and used by nodes to assess link burstiness over time. The acquired link burstiness information can be stored in the node's flash memory and relied upon to diagnose transmission problems when missed deadlines occur. Thus, accurate diagnosis is achieved in a distributed manner and without the overhead of transmitting rich measurement data to a central collection point. For the purpose of evaluation we have implemented BurstProbe in the GinMAC WSN protocol and we are able to demonstrate it is an accurate tool to debug time-critical data delivery. In addition, we analyze the cost of implementing BurstProbe and investigate its effectiveness.

1 Introduction

Future application areas for wireless sensor networks (WSNs) are industrial process automation and control systems. In such systems, the WSN is part of a control loop, and therefore predictable network performance in terms of message transfer delay and reliability is required.

WSNs for such applications are generally built around a schedule-based Medium Access Control (MAC) protocol. The schedule is calculated such that deadlines are met even when some retransmissions are necessary to compensate for losses on wireless links. Due to deadline and/or energy constraints it is obviously not possible to accommodate an arbitrary number of retransmissions and therefore a worst-case link reliability has to be assumed when determining a transmission schedule. Recently developed systems for time-critical data delivery such as WirelessHART [3], ISA100 [4], Munir's least-burst-routing [7], and GinMAC [13] use worst-case reliability assumptions when determining schedules. In WirelessHART a fixed number of redundant transmissions is used in the hope that these are sufficient to compensate for losses. Munir's least-burst-routing and GinMAC determine worst-case link reliability by measurement before deployment and determine the number of required retransmission slots on links based on this measurement.

Recent deployments show [7,13] that such provisioning before deployment is generally feasible, however, it cannot be guaranteed that link characteristics are invariant. For example, an interferer may appear (temporarily) in the vicinity of the network or links may become (temporarily) blocked by obstacles. It must be possible to either determine a new valid schedule or to identify and remove the problem source, which appeared in the sensor field post-deployment. It is necessary to collect appropriate debugging information during system operation allowing us to forensically investigate the problem that occurred.

It has been shown [7] that link quality in WSNs used for time-critical data delivery should be described with more precision than it is possible using simple metrics such as Packet Reception Rate (PRR) or Expected Transmission Count (ETX). To design efficient schedules for time-critical systems it is necessary to understand in detail the distribution and nature of burst errors on links. Unfortunately, as we show, it is impossible to gather such detailed link information by simply using existing data transmissions. Hence, it is very challenging to monitor link burstiness in a WSN deployment during network operation.

We propose to periodically measure link burstiness within the network in order to collect the necessary information for performance debugging in case that time-critical data delivery fails. Sequences of dedicated test transmissions - called BurstProbes - are used to obtain a clear picture of link burstiness over time. These probes are incorporated in the transmission schedule such that they do not interfere with the network's time-critical data delivery. As time-critical WSNs do not have sufficient spare capacity to continuously transmit all of their measurement data to a central collection point the data is stored in each node's flash memory. This data can then be retrieved during network maintenance after problems have occurred and network debugging is therefore required. The paper has the following specific contributions:

- BurstProbe: We introduce the novel concept of BurstProbe.
- BurstProbe Implementation: An implementation of BurstProbe within the GinMAC protocol and an experimental evaluation of its overhead and its effectiveness is presented.
- Debugging Examples: Different debugging examples using BurstProbe are presented. We show that problem sources can be identified and that new viable schedules can be calculated on the basis of recorded measurements.

We implement BurstProbe for GinMAC, a TDMA based MAC layer for time-critical data delivery that requires offline-dimensioning [13]. BurstProbe enables GinMAC to become adaptive by allowing it to determine the required number of retransmission slots when link characteristics change due to interference. BurstProbe is currently implemented in GinMAC, but it is suitable for use in any schedule-based WSN system aiming to achieve time-critical data delivery.

The paper is organised as follows. Section 2 introduces the BurstProbe concept. Section 3 provides a detailed description of the BurstProbe implementation

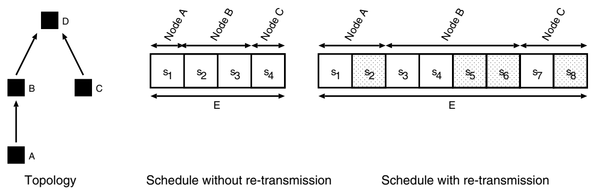

D
B
C
A
Node A
Node B
Node C
s1
s2
s3
s4
s5
s6
s7
s8
E
Topology
Schedule without re-transmission
Schedule with re-transmission

Fig. 1. Simple topology and possible schedules for error free and lossy channels

within GinMAC. Section 4 investigates the effectiveness of BurstProbes via experimental evaluation. Section 5 reports on related work dealing with the deployment of time-critical sensor networks. Section 6 concludes the paper.

2 BurstProbe

In this section we introduce the BurstProbe concept. We describe the motivation for its design and discuss its capabilities and limitations.

2.1 Scheduling for Timely and Reliable Data Delivery

Consider the simple network topology given in Fig. 1. Assume nodes A, B and C have to deliver data with period T to the sink node D. In order to guarantee timeliness a TDMA schedule is applied. A slot in the schedule accommodates the actual data transmission and a short acknowledgment from the receiver. In this paper, transmissions within a slot refer to both the original data packet and corresponding acknowledgment. If we assume that all nodes are in interference range of each other, the schedule as shown in Fig. 1 can be used. Node A transmits in slot s_1 to node B which uses s_2 to forward data from A to D; B uses slot s_3 to transmit its own data to D. Node C uses slot s_4 to transmit data to D. The resulting schedule $S = \{s_1, s_2, s_3, s_4\}$ has a duration (which we refer to as an *epoch*) of $E = |S| \cdot t = 4 \cdot t$ (with t being the slot length). Data from all nodes is delivered within the epoch to the sink. We refer to the schedule as *valid schedule* if it allows us to deliver data within the required period T. The schedule is valid if the epoch is shorter than the period ($E \leq T$).

Obviously, the schedule is only valid in situations where all transmissions are successful. In a wireless environment error free channels are rare and capacity for potential retransmissions must be incorporated within the schedule. Figure 1 shows a schedule for the aforementioned simple topology which allows for one retransmission on each link for each transmission. The epoch length has now doubled to allow for reliable and timely data delivery on potentially lossy links. The schedule is valid if $E \leq T$ and if no more than every second transmission is erroneous. Given the harsh radio environment where some sensor networks operate it is a challenge to provision the correct number of retransmission slots in advance.

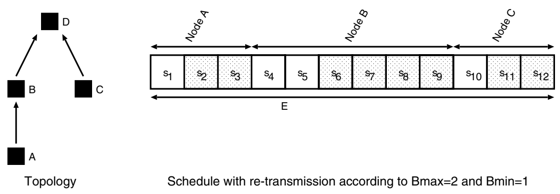

D
B
C
A
Topology
Node A
Node B
Node C
s1
s2
s3
s4
s5
s6
s7
s8
s9
s10
s11
s12
E
Schedule with re-transmission according to Bmax=2 and Bmin=1

Fig. 2. Topology and possible schedules for lossy channels with $B_{max} = 2$, $B_{min} = 1$

The epoch length can be reduced if all nodes are not within communication range of each other. In this case spatial re-use of TDMA slots is possible and the epoch can be shortened. However, it has to be noted that in industrial process automation and control scenarios in which time-critical scheduling is required it is common that all nodes are at least in interfering range of each other. In practical industrial deployments spatial slot re-use is rarely an option and systems such as GinMAC [13] and WirelessHART [3] do not make use of it. Instead, these systems support concurrent transmissions using several 802.15.4 channels to reduce epoch length.

2.2 Capturing Worst-Case Link Reliability

There are different methods available to describe link reliability. Common methods are Expected Transmission Count (ETX) or Packet Reception Rate (PRR). Using PRR gives a worst-case link reliability by a value P_{max} indicating that at least P_{max} transmissions out of n transmissions are successful. The problem with such a metric is that it does not capture the position of losses within the sequence of n transmissions. For example, the schedule allowing for retransmissions shown in Fig. 1 is not valid if transmissions in two or more successive TDMA slots fail. P_{max} might be large compared to n indicating a good quality link. However, this might not be entirely true if losses appear in bursts. Unfortunately, this is exactly what can be observed in practice: losses cluster [7].

It has been shown that burst lengths [7] are a much better metric to capture worst-case link reliability for networks that have to support time-critical data delivery. We define worst-case link reliability using the two values B_{max} and B_{min}: a link has no more than B_{max} consecutive transmission errors and provides at least B_{min} consecutive successful transmissions between two error bursts. It is possible to determine B_{max} and B_{min} before network deployment and to determine a schedule that can handle the *observed* worst-case [7,13]. Figure 2 shows the schedule for the example topology for $B_{max} = 2$ and $B_{min} = 1$. Again, this schedule can only be used if $E \leq T$.

2.3 Evaluating Worst-Case Link Reliability

During a deployment a schedule based on B_{max} and B_{min} may become invalid as channel conditions change for the worse. Likewise, a schedule may become inefficient as channel conditions improve. It is therefore desirable to track the development of B_{max} and B_{min} over time in order to be able to adapt the deployed schedule if necessary. Alternatively, it might be possible to identify and remove the cause of a link quality degradation.

Nodes could generally use the transmission slots that are assigned to them to test B_{max} and B_{min} in every epoch. However, most nodes within the network are not allocated sufficient transmission slots to determine an accurate reading of B_{max} and B_{min}. For example, nodes A and C in the topology shown in Fig. 2 have only 3 consecutive transmission slots available which would not allow detection of a change from $B_{max} = 2$ and $B_{min} = 1$ to $B_{max} = 3$ and $B_{min} = 1$.

To ensure that all nodes can accurately measure B_{max} and B_{min} it is necessary to assign them a sufficient number of consecutive slots within the TDMA schedule. In most scenarios it is not possible to supply all nodes in this manner because the TDMA epoch E would grow to exceed the time bound T required by the application.

2.4 BurstProbe

We propose to use dedicated probe slots to evaluate B_{max} and B_{min} during network operations. The usage of a set of probe slots is called BurstProbe. Probe slots are located at the end of the epoch within the TDMA schedule and are shared among nodes. Nodes are assigned temporary ownership of the probe slots which they subsequently use to measure link burstiness. A data packet and potentially a corresponding acknowledgment is transmitted in each probe slot and the probe sender records the success pattern. The allocation of probe slots to nodes can either be determined automatically or by a user that decides to gather data on specific links. Figure 3 shows the schedule for the example topology including 4 probe slots at the end of the schedule.

The measurement of B_{max} and B_{min} is carried out at a different point in time than a data transmission occurs. In the example shown in Fig. 3 node A transmits data in slots s_1, s_2 and s_3 at the beginning of an epoch, while link burstiness is measured in slots s_{13} to s_{14} at the end of an epoch. BurstProbe is only able to capture link burstiness if burst errors on a link are evenly distributed. However, our evaluation given in Sec. 4 shows that this is the case in real deployments and that BurstProbe is an effective measurement tool.

As probe slots are shared in the network, a node will not have access to them in every epoch (unless the node is assigned exclusive access). Hence, B_{max} and B_{min} is not tested in every TDMA epoch. However, link burstiness in practical deployments does not tend to change quickly but rather over many TDMA epochs. Thus, measurement of B_{max} and B_{min} in every other epoch is sufficient to obtain an accurate picture of link burstiness development over time. This is shown in the evaluation in Sec. 4 where we analyze real-world deployments using BurstProbe.

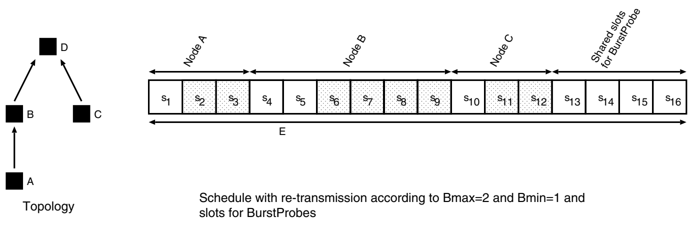

D
B
C
A
Topology
Node A
Node B
Node C
Shared slots for BurstProbe
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16
E
Schedule with re-transmission according to Bmax=2 and Bmin=1 and slots for BurstProbes

Fig. 3. Simple topology and possible schedules for lossy channels with $B_{max} = 2$ and $B_{min} = 1$ and 4 slots for BurstProbes

2.5 BurstProbe Effectiveness and Cost

The effectiveness of the BurstProbe mechanism is governed by the number of probe slots, the frequency with which probe sequences are executed and the nature of interference. Generally, the BurstProbe mechanism is more likely to determine an accurate B_{max} and B_{min} if a large number of probe slots are used and probe sequences are executed frequently. Infrequent usage of BurstProbe is feasible if the interference patterns are present for long periods.

The usage of BurstProbe causes additional energy costs. Firstly, a node must be active in additional slots to transmit and receive probe messages (Probing Cost). Secondly, the handling of the collected measurement data is energy costly as measurement data is stored locally (Storage Cost). Thus, the usage of Burst-Probe reduces nodes lifetime.

Our experiments (see Sec. 4) show that probing costs are significant. The duty cycle of a node further away from the sink doubles in realistic settings. However, it has to be noted that overall duty cycles are still very low. Storage costs in all cases are generally negligible.

2.6 BurstProbe Limitations and Optimisations

The outlined BurstProbe mechanism is only able to measure interference on a link properly if the interference occurs during the time the probes are executed. Strict periodic interference which falls in the transmission slots of a node but never within the probe slots cannot be detected. This limitation can be resolved by introducing a dynamic TDMA schedule where the exact schedule is calculated by all nodes at the start of an epoch. This would allow us to move probe slots to the location of the transmission slots. Essentially, the number of available transmission slots for a specific node would be temporarily increased to ensure that data transmission and probing occurs at the same point in time. Although such a mechanism could be implemented, the resulting system would be very complex. However, as our experiments in Sec. 4 show it is not necessary to implement BurstProbe in such a way; in typical deployments probing slots at the end of the schedule produce useful measurement results.

In addition, Burstprobe is designed for use in scenarios which have static or slow changing topologies. It does not provide any procedure to automatically distribute new schedules to nodes in a deployment (this type of procedure would need to be defined specifically for the TDMA system employed if required). However, it can be used to gather the data necessary to devise new schedules when needed.

3 BurstProbe Implementation

We implemented BurstProbe for GinMAC [13], a state of the art solution for time-critical data delivery in wireless sensor networks.

3.1 GinMAC

GinMAC is a wireless sensor networking MAC protocol that has been specifically designed to support time-critical application scenarios. Currently, GinMAC is deployed at an oil refinery in Sines, Portugal [12] where it is used to connect a number of sensors and actuators used to monitor and control product flow and processing. Nodes are grouped in small networks running GinMAC at different frequencies (called cells). Cells are interconnected using a wired backbone infrastructure. The use of cells ensures that GinMAC networks are relatively small, this is necessary to obtain short epochs E and tight delay bounds. GinMAC includes three main features of particular relevance to this task: Offline Dimensioning, Exclusive TDMA and Delay Conform Reliability Control. A network dimensioning process is carried out before the network is deployed. The input for the dimensioning process are network and application characteristics that are assumed to be known before deployment. The output of the dimensioning process is a TDMA schedule with epoch length E that each node has to implement.

The GinMAC TDMA epoch consists of three types of slots: *basic slots*, *additional slots* and *unused slots*. First, the epoch contains a number of *basic slots* which are selected such that within frame length E each sensor can forward i messages to the sink and the sink can transmit k messages to each actuator that might be present. Second, the GinMAC epoch uses *additional slots* to improve transmission reliability by providing capacity for retransmissions. Finally, the epoch may contain *unused slots* which are purely used to improve the duty cycle of nodes. The above types of slots within the GinMAC epoch must be designed such that the delay, reliability and energy requirements are met. However, it may not always be possible to find a schedule that simultaneously fulfils all requirements. The epoch E might be too long and thus application delay targets cannot be met. If that is the case, some dimensioning assumptions must be relaxed.

To facilitate the description of how the GinMAC protocol operates we provide here an example of how a simple wireless sensor networking topology is supported using GinMAC. Consider the deployment of the wireless sensor network topology with 7 nodes (including sink) depicted in Fig. 4. At deployment time the tree topology shown is determined to be feasible. All links are evaluated and $B_{max} = 1$ and $B_{min} = 1$ as worst-case on all links is determined.

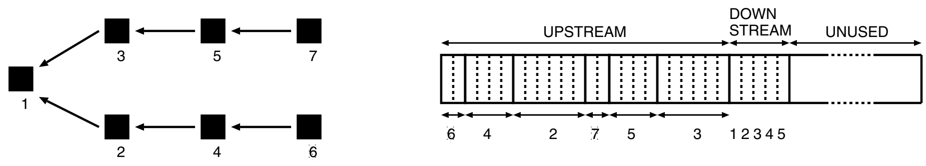


Fig. 4. 7 node example topology. $S_E = 100$ slots are used within the GinMAC epoch

Next, the delay requirement is obtained from the application; for this example we assume that all nodes must be able to transmit one message within $T = 1s$ to the sink node. Using a slot length of $10ms$ an epoch with $S_E = 100$ slots is feasible. Next the number of basic and additional slots can be computed. Slots are allocated starting from the leaf nodes. Node 6 is assigned the first 2 slots in the epoch; one for data and one for a potential retransmission. Node 4 is assigned the next 4 slots to accommodate transmission of data from node 6 and 4 including potential retransmissions. To accommodate transmissions and potential retransmissions a total of 24 slots are allocated for basic and additional slots to accommodate upstream traffic. Finally, 5 slots are necessary to allow one broadcast message to be forwarded from the sink to each node within an epoch. The downstream slots are used within GinMAC to perform time synchronisation of all nodes with the sink. Tight time synchronisation is necessary to implement an effective TDMA schedule. In the example, 29 out of 100 available slots are used for data transmissions. The remaining 71 slots are unused and can be used to optimise the nodes duty cycle and to grant application processes execution time. The resulting TDMA schedule is shown in Fig. 4.

The unused slots in GinMAC represent an ideal opportunity for introducing probing functionality into its transmission schedule without the risk of disrupting its primary data transmissions.

3.2 BurstProbes in GinMAC

Within the GinMAC protocol the most appropriate place to insert probes is the area of unused slots at the end of the active slots. We extended GinMAC such that a variable number of probe slots can be added at the end of the active slots. When BurstProbe is executed the result of the probe sequence is recorded in the node's flash memory.

The inclusion of probe slots within the schedule is not problematical. However, the recording of the BurstProbe measurement results on flash memory introduces a number of challenges. Nodes have a finite amount of flash memory which will be filled over time. At some point it is necessary to clear the used space to enable further writing. With flash memory entire sectors must be cleared first before they can subsequently be reused which requires relatively long sector clearance operations to be performed. The T-mote Sky, the mote used to execute the GinMAC implementation, has a flash capacity of 1MByte which is split into 16 sectors of size of 64KB, a sector erase cycle typically takes $1S$. Assuming only

50% of the flash is used to record probe data, 52,500 probe patterns could be written before such erasing cycles would be needed. In the configuration above this would occur after only approximately 102 hours of use and would occur many times over the life time of the network whilst still supporting data transmission. Therefore, it is essential to execute the blocking clearance operations such that the time-critical TDMA schedule is not disturbed.

4 Debugging GinMAC with BurstProbes

In this section we show that BurstProbe is a very useful tool to accurately monitor changes in link burstiness over time. These observations can be used to either refine the TDMA schedule or to identify and remove the interference source. We also measure the energy consumption of the BurstProbe mechanism.

4.1 Debugging Scenarios

The GinMAC protocol is designed for industrial process automation and control applications. For example, it is used to monitor production processes in an oil refinery [13]. In such a setting a number of typical events can be observed which have an impact on link quality within a deployed network. Typical events are:

1. Obstacle: An obstacle obstructs (temporarily) communication and link quality degrades. For example, a truck of a maintenance crew is parked temporarily within a communication link or a new production unit is installed obstructing communication.
2. Interference: Other wireless communication devices or machinery interferes (temporarily) with transmissions on links. For example, other networks or machinery such as pumps may interfere with communication.

The aim of BurstProbe is to identify and quantify these events such that a new TDMA schedule can be computed. In particular, we are interested in adjusting the number of retransmission slots as discussed in the previous section. Alternatively, if no valid schedule can be computed (due to severe link degradation) the aim of BurstProbe is to then provide debugging information to help identify and remove the source of link quality degradation. We evaluate the capability of BurstProbe to deal with both of these events.

For evaluation we use a simple network consisting of 7 nodes as shown in Fig. 4. For the experiments a schedule with $S_E = 100$ slots of length $10ms$ is used which results in an epoch time of $1s$. A variable number of transmission slots and probe slots are used in each of the executed experiments. Probe slots are assigned to nodes in a round robin fashion; each node carries out a probe sequence every 7 epochs. Probe results are recorded in the node's flash memory and can be analyzed offline. For the purpose of evaluating the BurstProbe mechanism each node also records the number of retransmissions required in each epoch.

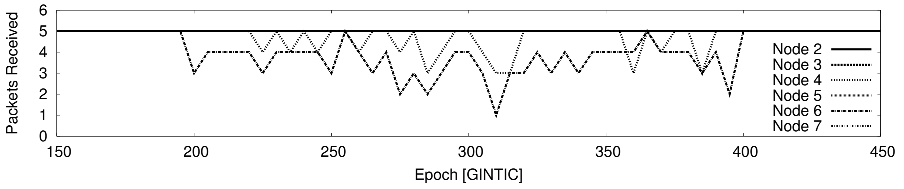
4.2 Interference on a Single Link

In the first experiment we configure the network for $B_{max} = 1$ and $B_{min} = 1$. As shown in Fig. 4 we need 29 transmission slots within the epoch of $S_E = 100$ slots. Initially, we use $S_P = 15$ slots for probing and the remaining 56 slots are unused. Each node transmits one data message within every epoch. In our experiments we run data transmissions for $t = 600s$. From $t = 200s$ to $t = 400s$ transmissions between node 4 and node 2 are disturbed. The disturbance is created by a purpose built interferer node that induces bursts of random size. Bursts have a worst-case characteristic of $B_{max} = 5$ and $B_{min} = 1$. Using an interferer node ensures that a ground truth can be established. The resulting link characteristic emulates a situation in which an interferer such as electric machinery or an obstacle would cause temporary link quality degradation.

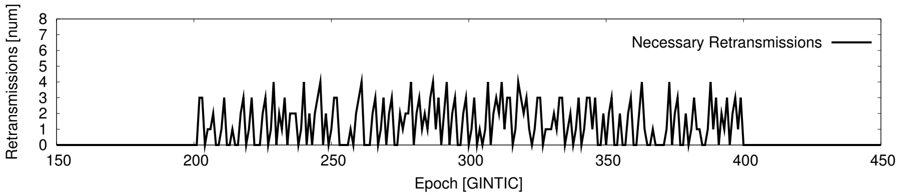
Figure 5 a) shows the message reception over time as recorded at the sink node. Every 5 epochs the number of messages received per node over the 5 epoch time period is plotted. At first all messages generated by all nodes are received at the sink. When the interference starts messages generated by node 4 and node 6 are lost. After the interference stops message losses return again to zero. By observing message arrival at the sink it is only possible to determine that a network problem was present from $t = 200s$ to $t = 400s$. However, it is not possible to infer from observations at the sink the location of the problem and how it could be cured. For this, better means of network debugging are necessary.

To gain more insight in the nature of the network problem we look at the necessary retransmission on links in the network. As messages have been lost some links must have used retransmission slots. The usage of retransmissions over time on the link between node 4 and 2 is shown in Fig. 5 b). Node 4 has 4 transmission slots available to transmit 2 messages every epoch to node 2. If more than 2 retransmissions are recorded within an epoch messages must have been lost. Node 6 is affected most by the link degradation as node 4 aims to deliver its own message first before forwarding messages of child nodes. The recordings of retransmissions on all other links in the network do not reveal retransmissions which shows that something must have interfered exclusively on the link between node 4 and node 2 at $200s < t < 400s$. However, even such a detailed recording of retransmissions over time does not help in determining how the TDMA schedule should be configured to deal with the problematic link. From the logs we can infer that the selected values for B_{max} and B_{min} on link $4-2$ are not correct. We cannot, however, infer the true value of B_{max} and B_{min} with this approach.

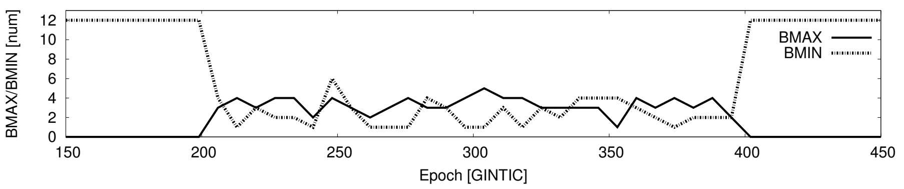
To obtain the true value of B_{max} and B_{min} we use BurstProbe. Figure 5 c) shows the recording of BurstProbe results obtained every 7 epochs on link $4-2$. Between epoch 200 and 250 B_{max} increases to a value of 4 while B_{min} drops to 2. The worst-case over the affected period is a $B_{max} = 5$ and $B_{min} = 1$. This measurement reflects the worst-case interference induced by the interfering node. With this data it is now possible to decide on corrective measures. A first option is to correct the schedule to include slots for the appropriate amount of potential



a) Message reception over time at the sink node



b) Retransmissions over time on link between node 4 and 2



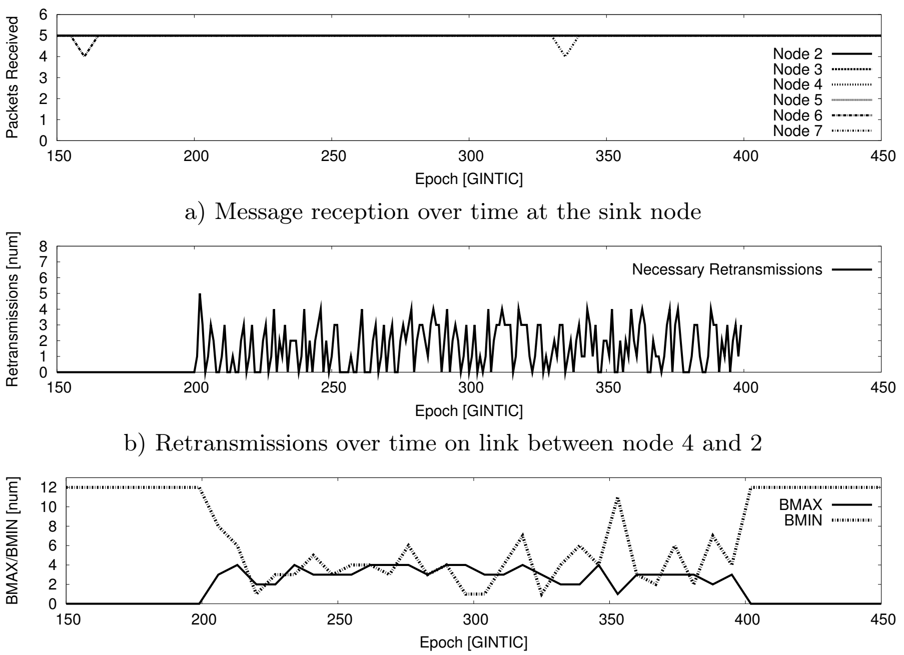
c) Recorded B_{min} and B_{max} values over time on link between node 4 and 2

Fig. 5. Message receptions over time at the sink; re-transmissions over time at link $4-2$; BurstProbe result for link $4-2$

retransmissions on link $4-2$. A second option would be to exclude the link from the topology if it is considered to be of a too poor quality. A third option would be to investigate the deployment to see if a potential interferer or an obstacle can be removed. For the purpose of this experiment, we decide to correct the TDMA schedule for link $4-2$ assuming a $B_{max} = 4$ and $B_{min} = 1$. As only one measurement showed a $B_{max} = 5$ we do not include it in the corrections.

We repeated our experiment after applying the corrections. Link $4-2$ now provides 10 transmission slots to handle $B_{max} = 4$ and $B_{min} = 1$. Fig. 6 a) shows that almost all messages are delivered to the sink. Figure 6 b) depicts that as expected 3 or 4 retransmissions are required. The probe results give the same indication on B_{max} and B_{min}. As we dimension for $B_{max} = 4$ and $B_{min} = 1$ the rare worst-case of $B_{max} = 5$ and $B_{min} = 1$ is not captured and some losses still occur. In summary, BurstProbe enables us to accurately determine a schedule that is able to handle the link quality degradation in the period $200s < t < 400s$.

We also measured the energy consumption of BurstProbe. First, we run the experiment with BurstProbe disabled and measure the nodes' transceiver usage time and flash usage time. Thereafter, we repeat the experiment with the BurstProbe mechanism. Our results show that the flash usage time for all nodes increases by approximately $20ms$ over the $600s$ experiment. This overhead is very



a) Message reception over time at the sink node

b) Retransmissions over time on link between node 4 and 2

c) Recorded B_{min} and B_{max} values over time on link between node 4 and 2

Fig. 6. Message reception over time at the sink; re-transmissions over time at link $4-2$; BurstProbe result for link $4-2$ using the re-provisioned epoch

Table 1. Radio Duty Cycle with and without BurtsProbes

Node	Standard	Probes	Increase
2	4.11%	5.34%	1.23%
4	3.02%	4.77%	1.75%
6	0.99%	2.87%	1.88%

small given that it is the equivalent of transmitting 4 additional packets over the 10 minute experiment. The changes in the transceiver duty cycle are shown in Tab 1. Based on the additional slots we expected an increase of 1.9% but the increase was slightly less due to traffic fluctuations caused by interference.

It is possible to decrease the energy consumption for the BurstProbe mechanism by reducing the number of probe slots. To investigate the effect of the number of probe slots on the accuracy of B_{min} and B_{max} we ran the experiment with $S_P = \{6, 8, 10, 12\}$ probe slots. The results are shown in Fig. 7. Only for $S_P = 6$ BurstProbe cannot identify the worst-case of $B_{max} = 5$ and $B_{min} = 1$. Thus, a probe number of $S_P = 8$ is sufficient for our application scenario. With $S_p = 8$, the energy cost for BurstProbe for node 4 decreases from 4.77% with $S_P = 15$ to 3.87%.

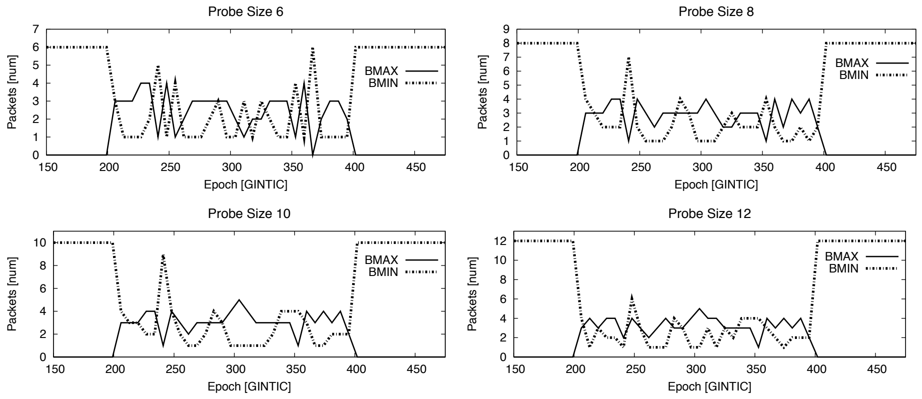


Fig. 7. B_{min} and B_{max} measurement with $S_P = \{6, 8, 10, 12\}$ probe slots

4.3 Network Wide Interference

In the second experiment we configure the network with the same initial configuration as in the first experiment in Sec. 4.2. The network has 7 nodes, using $B_{max} = 1$ and $B_{min} = 1$, with an epoch of size $S_E = 100$ slots and $S_P = 15$ probe slots. The experiment was deployed in the shape of an L, centred at the sink with each of the two branches running 90 degrees away from one another. As shown in Fig. 4 the first branch consisted of nodes 2, 4, 6 whilst the second had nodes 3, 5, 7. The network is used for $t = 600s$ to transmit data from each node to the sink at a rate of one packet per epoch per node. From $t = 200s$ to $t = 400s$ a Wi-Fi network occupying the same frequency as the network is enabled which generates interference. The Wi-Fi access point is located in the vicinity of node 3.

Figure 8 shows both the BurstProbe transmission results and the number of retransmissions recorded by each node. The figure is divided into two columns and three rows of smaller plots. The first column has the recorded values of branch one whilst the second has the values of branch two. With each row, the distance from the sink increases. The figure shows significant interference on links $2-1$, $3-1$, $5-3$ and $7-5$ where on a number of occasions all 15 probes were recorded as lost. At these points in time it is impossible to accurately calculate the values of B_{max} and B_{min}. The worst-case value for B_{max} and B_{min} where their value could be calculated was $B_{max} = 14$ and $B_{min} = 1$. This significant interference observed by the probe measurements was also seen in the high number of retransmissions recorded at each of the above links and high packet loss rate observed at the sink.

Due to the significance of the interference experienced, resolving the issues in a similar fashion to the first experiment is difficult. The worst-case values of B_{max} and B_{min} were not observed in the experiment as 15 probes proved insufficient. Furthermore, using the worst-case observable value of $B_{max} = 14$ and $B_{min} = 1$ to re-provision the network would require 197 transmission slots. This is more slots than are available within the epoch. Although the epoch size

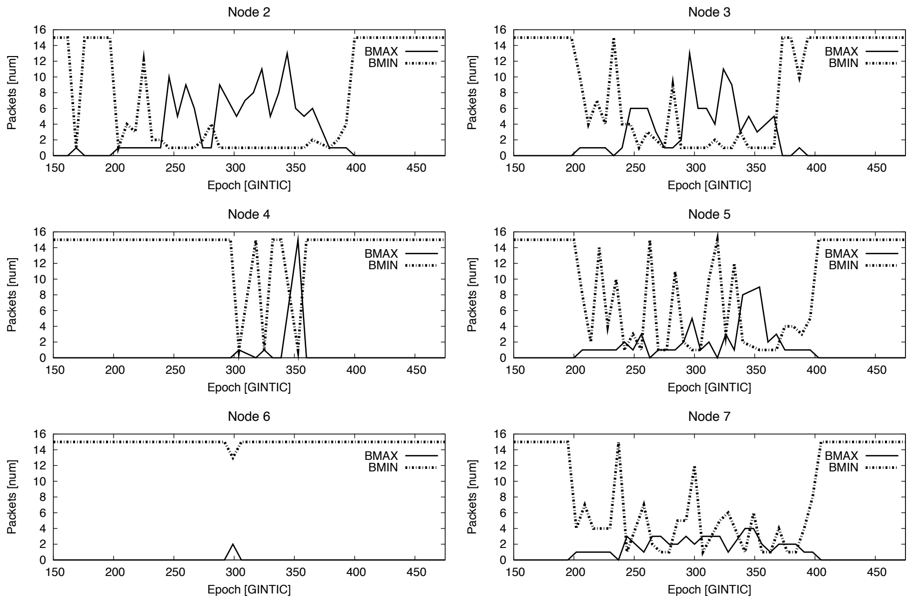


Fig. 8. Network wide interference caused by Wi-Fi

could be increased, as the epoch size increases so does the message delivery delay. Here the required epoch size would increase the communications delay beyond the value acceptable to the application. Therefore, simply re-provisioning the network is not a viable solution under the interference observed.

The second option to addressing interference issues, discussed in Sec. 4.2, is to remove problem links. The interference is widespread and occurs on the majority of links, simply removing links would not lead to a suitable solution. Therefore the only solution here is to identify and remove the source of the interference.

Figure 8 shows that the interference was recorded as being more prominent on links $2-1$, $3-1$, $5-3$ and $7-5$. This information could be used in a real deployment to identify the location of the interference source. Examining Fig. 8 we can deduce that the interference source must be in the vicinity of the sink and the first branch as the interference in the second branch reduces with distance from the sink. This would provide valuable information in pinpointing the location of the interference to eradicate it. These results confirm the actual location of the Wi-Fi interference source as in the vicinity of node 3.

5 Related Work

To date WSN research has produced a number of solutions aimed at addressing timely data delivery in wireless sensor networks [7,3,4,13]. All of these solutions require precise knowledge of available link quality in order to select transmission schedules. Thus, the BurstProbe mechanism outlined in this paper may be applied to any of these solutions. With regards to the BurstProbe approach and the task of debugging wireless sensor networks in general, earlier techniques used for analysing performance problems relied on the data sink retrieving live debug data from each node in the network [10]. Other techniques used additional nodes to monitor radio traffic and problems [1] [11]. The BurstProbe approach is based on the concept of inserting dedicated probes into unused transmission slots and recording the probe result patterns locally. Other examples where performance data is stored locally on motes include [8] where an approach for diagnosing performance anomalies is presented that highlights the potential benefits of using local flash storage of system data, and Envirolog [6] that is designed to allow the user to produce an exact replay of recorded events and conditions to aide performance evaluation. Also related are [5] and [9] which embed performance related data in application messages to employ a passive approach to anomaly detection. PD2 [2] takes an alternative approach by focusing on data flows that individual applications generate, relating poor application performance to loss and latencies of data flows. This allows performance monitoring and debugging information to only be enabled on the nodes that data flows are known to traverse, thus reducing the overall overheads imposed. However, whilst all of these proposed approaches offer different techniques and models for recording and in some cases disseminating performance information, BurstProbe provides novelty by provisioning specific transmission slots to insert dedicated probing that can help determine more accurate information about loss and retransmissions that are occurring throughout a wireless sensor network deployment.

6 Conclusion

In this paper we presented BurstProbe, a mechanism useful to debug time-critical WSNs. As shown, BurstProbe is a useful tool for measuring changes in link burstiness over time within a running network. The BurstProbe measurements can be used to correct TDMA schedules or to locate sources of interference. A key characteristic of the BurstProbe mechanism is that it can be included in WSN systems such as GinMAC or WirelessHART without interfering with the time-critical data delivery. The cost of the BurstProbe mechanism in terms of energy cannot be neglected; in the described experiments node duty cycles increase in the order of 2%. However, we believe that such relatively modest energy investment is necessary in order to be able to debug WSN for time-critical data delivery. In our prototype deployment at an oil refinery in Sines, Portugal the resulting node lifetime of a few months is acceptable as default system maintenance is carried out frequently. In the paper we discussed the basic functionality of BurstProbe but many optimisations and refinements are not explored. Firstly, it is necessary to investigate scheduling mechanisms for probe slots. It would be useful to schedule more probes on links that currently experience problems while reducing probe frequency on good links. Secondly, it would be useful to implement a mechanism to fetch recorded burst probe data remotely rather than collecting nodes and extracting the data from the flash manually for analysis. A mechanism as described in [9] might be a good starting point for such extension.

Acknowledgments. This work has been partially supported by the EC under the FP7 contract ICT-224282 (GINSENG) and the FP7 contract ICT-224053 (CONET).

References

1. Chen, B., Peterson, G., Mainland, G., Welsh, M.: LiveNet: Using passive monitoring to reconstruct sensor network dynamics. In: Nikoletseas, S.E., Chlebus, B.S., Johnson, D.B., Krishnamachari, B. (eds.) DCOSS 2008. LNCS, vol. 5067, pp. 79–98. Springer, Heidelberg (2008)
2. Chen, Z., Shin, K.G.: Post-deployment performance debugging in wireless sensor networks. In: Proceedings of the 2009 30th IEEE Real-Time Systems Symposium, RTSS 2009 (2009)
3. HART Communication Foundation: Wirelesshart data sheet (2010), `http://www.hartcomm.org/`
4. International Society of Automation (ISA): Isa100 wireless systems for automation (2010), `http://www.isa.org/`
5. Liu, K., Li, M., Liu, Y., Li, M., Guo, Z., Hong, F.: Passive diagnosis for wireless sensor networks. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, SenSys 2008 (2008)
6. Luo, L., He, T., Zhou, G., Gu, L., Abdelzaher, T.F., Stankovic, J.A.: Achieving repeatability of asynchronous events in wireless sensor networks with envirolog. In: Proceedings of the IEEE Conference on Computer Communications, INFOCOM 2006 (2006)
7. Munir, S., Lin, S., Hoque, E., Nirjon, S.M.S., Stankovic, J.A., Whitehouse, K.: Addressing burstiness for reliable communication and latency bound generation in wireless sensor networks. In: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN 2010 (2010)
8. O'Donovan, T., Tsiftes, N., He, Z., Voigt, T., Sreenan, C.J.: Detailed diagnosis of performance anomalies in sensornets. In: Proceedings of the ACM Workshop on Hot Topics in Embedded Networked Sensors, HOTEMNETS 2010 (2010)
9. Pejovic, V., Sreenan, C.J.: Perdb: Performance debugging for wireless sensor networks. In: Proceedings of the European Conference on Wireless Sensor Networks, Poster/Demo session, EWSN 2009 (2009)
10. Ramanathan, N., Chang, K., Kapur, R., Girod, L., Kohler, E., Estrin, D.: Sympathy for the sensor network debugger. In: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, SenSys 2005 (2005)
11. Ringwald, M., Römer, K., Vitaletti, A.: Passive inspection of sensor networks. In: Aspnes, J., Scheideler, C., Arora, A., Madden, S. (eds.) DCOSS 2007. LNCS, vol. 4549, pp. 205–222. Springer, Heidelberg (2007)
12. Sreenan, C.J., SaSilva, J., Wolf, L., Eiras, R., Voigt, T., Roedig, U., Vassiliou, V., Hackenbroich, G.: Performance control in wireless sensor networks: the ginseng project - [Global communications news letter]. Communications Magazine 47(8) (August 2009)
13. Suriyachai, P., Brown, J., Roedig, U.: Time-critical data delivery in wireless sensor networks. In: Rajaraman, R., Moscibroda, T., Dunkels, A., Scaglione, A. (eds.) DCOSS 2010. LNCS, vol. 6131, pp. 216–229. Springer, Heidelberg (2010)

The Announcement Layer: Beacon Coordination for the Sensornet Stack

Adam Dunkels, Luca Mottola, Nicolas Tsiftes, Fredrik Österlind, Joakim Eriksson, and Niclas Finne

Swedish Institute of Computer Science
{adam,luca,nvt,fros,joakime,nfi}@sics.se

Abstract. Sensornet protocols periodically broadcast beacons for neighborhood information advertisement, but beacon transmissions are costly when power-saving radio duty cycling mechanisms are used. We show that piggybacking multiple beacons in a single transmission significantly reduces transmission costs and argue that this shows the need for a new layer in the sensornet stack—an announcement layer—that coordinates beacons across upper layer protocols. An announcement layer piggybacks beacons and coordinates their transmission so that the total number of transmissions is reduced. With an announcement layer, new or mobile nodes can quickly gather announcement information from all neighbors and all protocols by issuing an announcement pull operation. Likewise, protocols can quickly disseminate new announcement information to all neighbors by issuing an announcement push operation. We have implemented an announcement layer in the Contiki operating system and three data collection and dissemination protocols on top of the announcement layer. We show that beacon coordination both improves protocol performance and reduces power consumption.

1 Introduction

Sensor network protocols use periodic beacons to advertise information to neighbors. Examples include routing cost gradients in data collection protocols [11,23,24], version or sequence numbers in data dissemination protocols [12,15,16], and presence information in neighbor discovery protocols [8,14]. Beacons are transmitted both periodically and when protocols detect potential inconsistencies. For example, a node in a collection protocol that detects a loop repairs the network by asking its neighbors for the latest routing cost gradient [11], and a node in a dissemination protocol that has an older version than its neighbors achieves consistency by asking its neighbors for the latest version [15].

Beacons are transmitted as broadcasts so that they reach all nodes in the neighborhood of the transmitter. Broadcast are, however, costly in terms of power since low-power networks duty cycle their radios. Many protocols therefore attempt to reduce the amount of beacons they transmit. For example, the CTP data collection protocol uses adaptive beaconing [11] and the Trickle and the

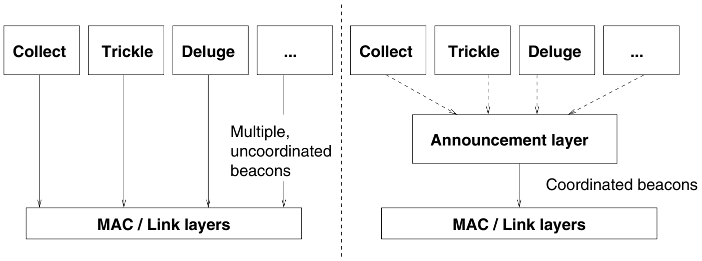


Fig. 1. The announcement layer coordinates beacons from multiple protocols. This allows the number of beacon transmissions to be reduced and announcement operations to be coordinated across protocols.

RPL protocols uses beacon suppression [15,23]. These solutions only work for one individual protocol, however. When multiple protocols are used concurrently, each protocol will transmit their beacons independently of each other, increasing the power consumption.

We argue that there are significant power savings to be made through coordinating and piggybacking beacons from multiple protocols. We present the announcement layer, a beacon coordination layer for the sensornet stack that transmission of periodic beacons from multiple protocols, piggyback multiple beacons into each transmission to reduce the total amount of beacon transmissions, and provide operations for pushing announcements to the neighborhood and for requesting announcements from neighbors. Figure 1 shows how the announcement layer fit into the network stack.

The announcement layer defines two operations: *push* and *pull*. Push quickly transmit an announcement to the neighborhood and pull requests announcements from all neighbors. The push operation is used e.g. when a collection protocol finds a better route and the pull operation is used e.g. when a collection protocol node detects a routing loop and the latest routing gradients are needed.

We argue that an announcement layer provide at least three benefits. *Reduction of bandwidth usage and network congestion* since multiple beacons are collected in a single broadcast transmission (Section 5.1). *Reduction of power consumption* since the marginal cost of sending larger packets is low (Section 5). *Easier inter-protocol coordination* since announcements from multiple concurrent protocols are collected at a single point in the network stack, push and pull operations can be made across protocols (Section 5.2).

We make our case as follows. We demonstrate that beacon transmissions are costly that multiple transmissions even more so (Section 2). This motivates the need for an announcement layer (Section 3). We have implemented an announcement layer in Contiki (Section 4) and demonstrate that the use of an announcement layer reduces the number of beacon transmissions and the power consumption in a series of simulations and testbed experiments (Section 5) and discuss how an announcement layer differs from existing approaches (Section 6).

2 Motivation: Beacon Transmissions Are Costly

The background to the announcements programming abstraction stems from the periodic broadcast beacons used by sensor network protocols to do one-hop neighborhood information advertisement, and the observation that radio duty cycling makes broadcast expensive.

2.1 Sensornet Protocols Use Beacons

Sensor network protocols use periodic beacon transmissions to advertise information to the one-hop neighborhood. Examples include route metrics in data collection protocols [11] and version numbers in data dissemination protocols [12].

Information advertisement within the physical neighborhood of sensor nodes may also serve functionality reaching up to the application level, e.g., to coordinate sensors and actuators in a control application [5]. Programming systems for such application-level information sharing exist, such as Hood [22] and TeenyLime [4].

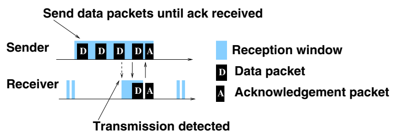
Beacons are typically transmitted periodically, but the period often changes over time. Many protocols exponentially increase their beacon rate when the information in the beacons has been transmitted several times and is no longer new. Examples include the Trickle single-packet data dissemination protocol [15], the multi-packet data dissemination protocol Deluge [12], the CTP data collection protocol [11], and the RPL low-power IPv6 routing protocol [23].

2.2 Duty Cycling Makes Beacon Transmissions Costly

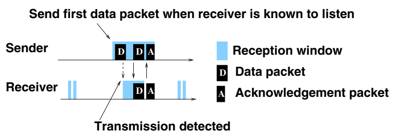
Beacons need to reach all nodes in the neighborhood of the transmitting node. Beacons are therefore sent as broadcast messages, but since radio duty cycling must be used to maintain a low power consumption, broadcast messages become comparatively expensive in terms of power consumption.

To send a broadcast transmission, the sender must make sure that all its neighbors are awake to receive the broadcast transmission. Ensuring that all neighbors are awake to receive the transmission can be done in two ways: either by having all nodes agree on scheduled rendez-vous when all nodes are simultaneously awake, or by having the sender explicitly wake up all its neighbors. Scheduled rendez-vous are costly since nodes must wake up for every rendez-vous, even if no data is to be transmitted. Explicit wake-ups are expensive since the sender must make sure that all nodes receive the message, thus typically needs to transmit the message multiple times. In contrast, for a unicast transmission, it is enough that only one node—the receiver—is awake to receive the message. Thus unicast transmissions are fundamentally less expensive.

We perform a set of experiments to quantify the power cost of broadcast transmissions. We use Contiki 2.5 with the ContikiMAC duty cycling protocol, the default duty cycling protocol in Contiki 2.5. ContikiMAC is a low-power-listening MAC protocol that builds on mechanisms from many existing state-of-the-art duty cycling protocols [2,10,17,19] but adds a very power-efficient channel



(a) Low-power listening with ContikiMAC.



(b) Transmission phase-lock.

Fig. 2. After a successful transmission, the sender has learned the channel sampling phase of the receiver, and subsequently needs to send only two transmissions

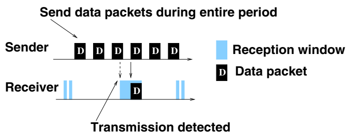


Fig. 3. A broadcast transmission must wake up all neighbors. The sender therefore extends the packet train for a full channel sampling period.

sampling mechanism. From B-MAC [19], ContikiMAC lends the basic idea of low-power listening. From X-MAC [2], ContikiMAC uses the idea of a packetized preamble. From WiseMAC [10], ContikiMAC uses the phase-lock mechanism, which we describe below. From BoX-MAC [17], ContikiMAC uses the idea of using the data packet as the wake-up signal.

Figure 2 shows the basic operation of ContikiMAC. Nodes wake up periodically to sample the radio medium for transmissions. This is performed in a power-efficient way: a node turns the radio on for only 192 microseconds to measure the received signal strength. If this indicates a transmission from a neighbor, the node keeps the radio on. To avoid missing transmissions, the node samples the radio medium twice within 0.5 ms. A sender triggers a transmission by sending a train of data packets, until one packet finds the receiver's radio on. Upon receiving a packet, the receiver answers with a link-layer acknowledgment and the sender stops transmitting the packet train. In ContikiMAC, unicast transmissions are power-efficient because senders phase-lock to the wake-up interval of its neighbors [10]. A sender synchronizes to the wake-up phase after the successful transmission, as shown in Fig. 2. For broadcasts, the sender needs to send its packet train for a full channel sampling period to ensure that all neighbors have heard the transmission, as shown in Fig. 3.

To quantify the relative cost of broadcast and unicast, we perform an experiment using two TMote Sky motes running Contiki and ContikiMAC. We use a channel sampling rate of 16 Hz. We run two experiments, one where we send broadcast traffic and one where we send unicast traffic, and vary the send rate. We measure the radio duty cycle using Contiki's built-in software-based power profiler [6]. Figure 4 shows the results. We observe that the cost of broadcast is significantly higher than that of unicast, and that the marginal increase in power

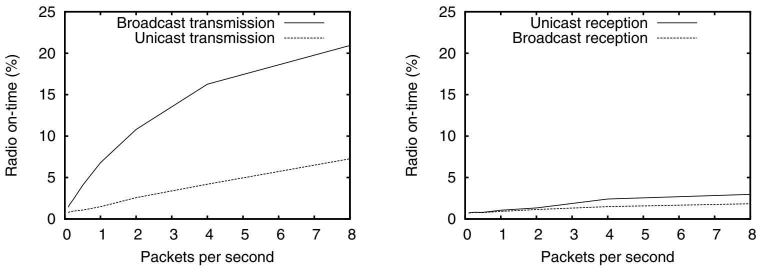


Fig. 4. The power consumption of broadcast is significantly higher than for unicast because broadcast transmissions (Fig. 3) are longer than unicast transmissions (Fig. 2)

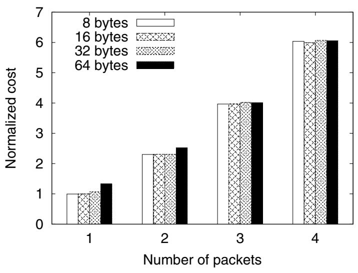


Fig. 5. Multiple, small broadcast transmissions are significantly more costly than a single broadcast transmission, for the same amount of data

consumption with increasing send rate is higher for broadcast. Therefore, there is much larger room for improvement in optimizing broadcast transmissions rather than unicast transmissions.

2.3 Multiple Transmissions Are More Costly

We perform another experiment, using the setup as above, where we transmit a fixed amount of data split into one, two, three, or four broadcast transmissions. The total amount of data is the same in all four cases, and we vary the amount of data across experiments. The purpose is to study the power consumption of multiple transmissions versus that of a single transmission, containing the same amount of data.

The result is shown in Fig. 5 and shows that multiple transmissions are significantly more costly than a single transmission, with the same amount of data. Also, the marginal cost of transmitting additional bytes in a single broadcast is significantly lower than the cost of transmitting the data as multiple transmissions. This demonstrates that there is a strong incentive to reduce the number of broadcast transmissions by collecting multiple beacons into a single, larger packet.

3 The Announcement Layer

The announcement layer provides beacon coordination for upper layer protocols. Protocols register announcements with the announcement layer and the announcement layer takes care of the periodic transmission of the announcements. An announcement is the information that a protocol would otherwise periodically transmit as beacons.

The announcement layer piggybacks announcements from multiple protocols in each beacon transmission and coordinates the transmissions so that the total amount of transmissions is reduced. Since the information sent in each announcement is typically small [11,15,16], several announcements often fit in a single beacon.

In addition to beacon coordination, the announcement layer provide a small but powerful set of operations that give protocols the ability to push announcements to neighbors and to pull announcements from neighbors.

An announcement is a key-value pair. The key is an integer that uniquely identifies the announcement. The value is a data array. The semantics of the value in an announcement is application-specific and opaque to the announcement mechanism.

Each announcement has a minimum rate for its periodic transmissions. The rate is set by the protocol that registered the announcement, and can be different for different announcements. From the minimum rates set for each announcement, the announcement layer computes a schedule that ensures that one and only one beacon is transmitted for every beacon interval. Since multiple announcements are consolidated into each beacon transmission, it is enough to send one beacon for each interval, thus reducing the total amount of beacon transmissions. This also means that an announcement may be transmitted more often than its minimum rate, which is allowed by the semantics of the announcements layer because protocols specify only the minimum transmission rate, not the maximum rate.

Each announcement has a scope that is either node scope or network scope. Node-scope announcements have a value that is bound to the node, whereas network-scope announcements have a value that is shared across the network. An example of a node-scope announcement is a hops-to-sink metric in a data collection protocol, which is specific to the node that registered the announcement. An example of a network-scope announcement is a global version number in a data dissemination protocol, which is the same for all nodes in the network.

3.1 Beacon Coordination

The announcement layer coordinates beacon transmissions for all registered announcements. The values from multiple beacons is collected into a single broadcast transmission. The beacon intervals from all protocols are coordinated to reduce the total amount of beacon transmissions.

Each protocol registers its maximal beacon interval with the announcement layer. The announcement layer ensures that at least one beacon is transmitted

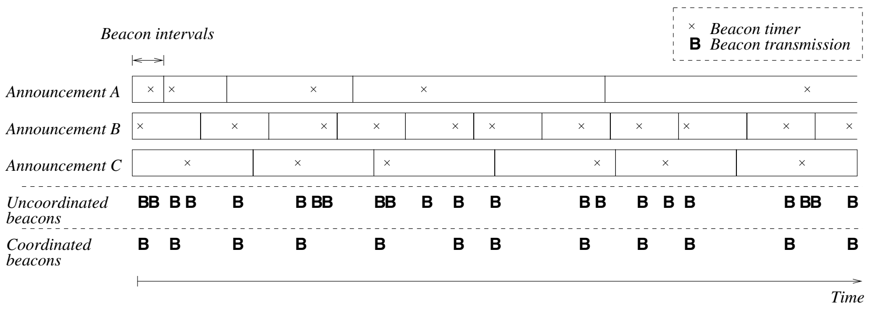


Fig. 6. Beacon coordination: Protocols A, B, and C have registered announcements A, B, and C. Data from all announcements are consolidated into each beacon transmission. With beacon coordination, only one beacon per announcement interval is transmitted. In this example, beacon coordination reduces the number of beacon transmissions from 22 to 12.

within this time interval. If two or more protocols need to transmit a beacon within a given time interval, beacon coordination will see to it that only the first beacon is sent. Since each beacon contains all announcements, the first transmission is enough to ensure that both announcements are transmitted within their respective intervals.

The beacon coordination concept is illustrated in Fig. 6. In the illustration, three protocols have registered three announcements. Each announcement has a different beacon interval. A beacon is to be sent randomly within each interval. Without beacon coordination and beacon consolidation, each protocol would send their own beacon messages without coordinating with the other protocols and the system in Fig. 6 would transmit 22 beacons. By contrast, with beacon coordination and beacon consolidation, the system sends only 12 beacons.

The beacon coordination algorithm is simple. For each new announcement, a timer fires randomly within each interval. When the timer fires, it checks if a beacon has already been transmitted within its interval. Unless the protocol has updated the value of its announcement since the last beacon transmission, there is no need to send a new beacon and the transmission is consequently cancelled.

3.2 Announcement Operations

The announcement layer defines two primary operations, *push* and *pull*. In addition, the announcements layer provides functions for registering announcement, to set the value of an announcement, and to set the minimum rate for the periodic transmission of announcements. Figure 7 shows the operations and functions.

Protocols that use announcements first register the announcement with the *register* function. The registration is a simple procedure that binds a key to the announcement and sets its scope. After registering the announcement, the protocol will begin receiving notifications when neighbors transmit their announcements. This is handled via a callback function that is given as an argument to

`push(key)`	`Pushes an announcement to neighbors.`
`pull(key)`	`Pulls an announcement from neighbors.`
`register(key, scope, callback)`	`Registers an announcement and set its scope.`
`setValue(key, value)`	`Sets the value of an announcement.`
`setMinRate(key, rate)`	`Sets the min periodic transmission rate.`

Fig. 7. Announcement layer operations

the *register* function. The protocol gives the announcement its value with the *setValue* function. The minimum transmission rate of the announcement is set with the *setMinRate* function.

The *push* and *pull* operations are used for pushing announcements to the neighborhood and to request announcements from neighbors, respectively. A *push* causes the transmission of an announcement to all neighbors. Protocols can use this operation to send new information to neighbors quickly. For example, a node in a data collection protocol that learns a significantly better route may want to quickly let its neighbors know of this new route. The node would set a new value for its announcement with the *setValue* function, and then call the *push* operation to push the changed routing metric to its neighbors.

The *pull* operation requests announcements from neighbors. The operation causes one or more neighbors to send their announcements to the node that issued the pull. This operation is used by protocols that are able to discover when a node needs information from its neighbors. For example, when a mobile node moves into a new environment it needs to gather information about its network environment, such as routing metrics. It then issues a *pull*, which causes its neighbors to send their announcements to the requesting node.

Although both the push and pull operations are defined to operate only on a single announcement, the beacon consolidation will collect all a node's announcements the beacon transmission. This means that a single push operation will push all announcements to neighbors. Likewise, a single pull operation will pull all announcements from neighbors.

The push and pull operations are based on the observed behavior of existing protocols:

Data collection with CTP. The CTP data collection protocol [11] defines an implicit push operation and an explicit pull operation. When the routing metric of a node changes significantly, CTP quickly transmits a beacon message with the new routing metric, which constitutes an implicit push operation. This makes the new information propagate faster than with the usual periodic dissemination. If a node detects that its routing metric is out of date, e.g., if a loop is detected or if the node has just started, CTP sends a beacon with a pull-bit set, to which neighbors respond by sending a beacon. This is an explicit pull operation.

Single-packet data dissemination with Trickle. The Trickle single-packet dissemination protocol [15] defines implicit push and pull operations. When Trickle starts sending a new version of the data to be disseminated, Trickle nodes issue an implicit push to rapidly propagate the new version through the network. Also, if a node notices that a neighbor has an older version than the current global version, the node performs an implicit push by directly broadcasting a beacon with the latest version. When a node boots up, it performs an implicit pull operation by sending a beacon with version number zero, which causes neighbors to broadcast the latest version.

Low-power IPv6 routing with RPL. The RPL IPv6 routing protocol defines explicit push and pull operations [21,23]. Nodes periodically transmit DODAG Information Objects (DIO) messages to let nodes in their one-hop neighborhood discover and maintain routes. The DIOs are transmitted at adaptive intervals and RPL also uses suppression to reduce the amount of transmissions. The combined effect of adaptive intervals and suppression may cause a node that has just started or moved into a new neighborhood to wait for a long time before getting a DIO. An RPL node may therefore send a DODAG Information Solicitation (DIS) message, to which neighbors reply with DIO messages. This constitutes an explicit pull.

Neighbor discovery. Many neighbor discovery protocols for mobile sensor networks [8,9,13,25] define both push and pull operations, which may be triggered by physical mobility. Nodes transmit beacons to announce their presence, constituting an explicit push, and may request information from their neighborhood to gather the identity of surrounding devices when the connectivity may change because of a changing physical location.

3.3 Protocol Implementations with an Announcement Layer

To demonstrate the feasibility of the announcement layer as a programming primitive for network protocols, we have rewritten three of the most common sensor network protocols to use announcements: data collection, single-packet data dissemination, and multi-packet data dissemination. All three protocol implementations are based on the original implementations in Contiki [7]. The data collection protocol is Contiki collect, an address-free, tree-based collection protocol similar to the TinyOS Collection Tree Protocol [11]. The single-packet data dissemination protocol is based on the Trickle protocol by Levis et al [15]. The multi-packet data dissemination protocol is Deluge [12]. We describe the implementation of the data collection protocol in detail below.

The starting point for our announcements-based data collection protocol is the Contiki collect protocol. Contiki collect builds a tree, rooted at the sink, by letting each node estimate the expected number of transmissions (ETX) to reach the sink. Nodes outside the neighborhood of the sink select the neighbor with the lowest ETX routing cost as their parent in the tree. Each node announces its ETX value to its neighbors through periodic beacons. The beacons are transmitted with an increasing interval, but when a node finds a significantly better parent, the beacon interval is reset to a low value.

The original Contiki collect module [7] uses periodic beacons to advertise routing cost. In our rewritten variant, the protocol instead uses announcements to advertise its routing cost. An excerpt of the rewritten version is shown as

```
collectInit() {
  register(COLLECT_KEY, NODE_SCOPE)
  pull(COLLECT_KEY)
}
receivedAnnouncement(fromAddress, etx) {
  addNeighbor(fromAddress, etx)
  updateLocalETX()
  setValue(COLLECT_KEY, localETX)
  if (newParent) {
    push(COLLECT_KEY)
    setMinRate(COLLECT_KEY, lowestRate)
  }
}
sendDataPacket() {
  if (parent == nil) {
    pull(COLLECT_KEY)
  } else {
    sendto(parent)
  }
}
```

Fig. 8. The relevant parts of the data collection protocol with announcements, in pseudo code

pseudo code in Fig. 8. When the collection protocol is initiated, it registers an announcement with a pre-defined key and with the node scope. This announcement is used for advertising route metrics. When the node starts, it does not have any route information and therefore issues a *pull* to get route information from neighbors. Likewise, when a node has no parent, it performs a *pull* to obtain one. When a new parent has been found, the node does a *push* to let others know about its new route.

4 Implementation

We have implemented an announcement layer in the Contiki operating system and the Rime network stack [7]. The announcement layer is implemented as a separate Rime module that uses a Rime broadcast channel to send and receive its beacon messages.

Beacon coordination consolidates all announcements into each beacon packet, but technology-specific limitations on radio packet size may restrict the amount of announcements that can be consolidated into each packet. For example, the popular 802.15.4-2006 standard defines a maximum packet size of 127 bytes[1]. Our announcement layer implementation handles this by breaking up large beacons into multiple broadcast transmissions.

To avoid instantaneous congestion caused by network synchronization, our implementations of the *push* and *pull* operations incur a random wait period before the beacons are transmitted. In our implementation, we set the waiting period to a random time between 0 and 8 seconds.

[1] Upcoming versions of the standard increase the maximum packet size to 2047 bytes.

5 Evaluation

We evaluate three aspects of the announcement layer. First, we quantify the beacon coordination mechanism and the resulting reduction in power consumption. Second, we quantify the cost of the announcement operation in terms of power consumption and the number of packet transmissions. Third, we use a testbed experiment to study a sensor network with concurrent protocols.

We use both simulation and testbed experiments. All simulations and experiments are carried out with Tmote Sky motes. For our simulations, we use the Contiki Cooja network simulator and the MSPsim Tmote Sky emulator [20]. Cooja and MSPsim provides a cycle-level accurate emulation of the MSP430 microcontroller and a bit-level accurate emulation of the CC2420 radio transceiver, which makes it possible to correctly emulate low-level protocols such as radio duty cycling mechanisms. For our testbed experiments, we use a 24-node Tmote Sky testbed in an office environment with a wired backchannel through which we obtain logging information. Throughout our experiments, we use Contiki 2.5 and the ContikiMAC low-power listening duty cycling mechanism with a channel check rate of 8 Hz, which results in an idle duty cycle of 0.5%.

We use the radio duty cycle as a proxy for energy consumption because the radio transceiver is the most power consuming component. We use Contiki's power profiler to measure the radio duty cycle [6], both the amount of time that the radio spends in listen mode and in transmit mode.

5.1 Beacon Coordination

The purpose of beacon coordination is to reduce the number of beacon transmissions by consolidating all announcements into every beacon and by suppressing the transmission of redundant beacons.

To evaluate the effectiveness of the beacon coordination mechanism, we set up a system with a variable number of announcements and set a fixed minimum rate of ten seconds for each announcement. We vary the number of announcements and measure the number of beacons that get transmitted as well as the total power consumption of the system. We run the system both with and without beacon coordination.

Figure 9 shows the result. We see that without beacon coordination, the number of beacons per interval increases with the increasing number of announcements. With beacon coordination, however, the number of beacons remains at one per interval. Similarly, without beacon coordination, the power consumption grows with the number of announcements, but with beacon coordination the power consumption stays almost constant, even though there is a slight increase in power consumption due to the additional size of each beacon.

5.2 The Cost of Announcement Operations

The push and pull announcement operations involve the transmission and reception of network traffic, incurring power consumption in the involved nodes. We quantify the effect of these operations on the power consumption by conducting

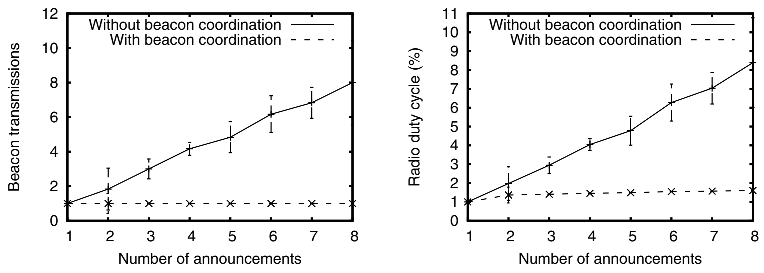
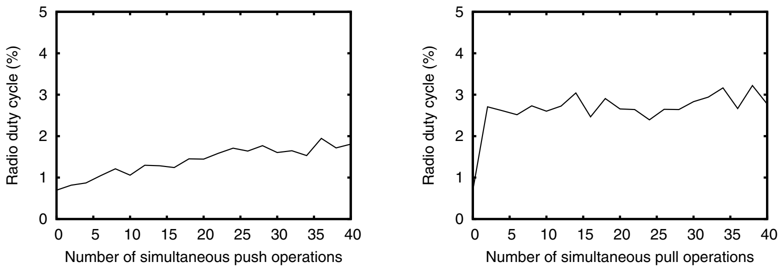


Fig. 9. The number of beacon transmissions with and without beacon coordination (left). The power consumption with and without beacon coordination (right).



(a) The cost of push (b) The cost of pull

Fig. 10. The cost of the push and pull operations in a 40 node network where all nodes are in range of each other

three experiments. First, we measure the effect on power consumption caused by the push operation in a dense network with a varying number of nodes that issue a push operation. Second, we measure the effect of the pull operation in the same situation. Third, we quantify the marginal cost of an increasing number of announcements being pushed in a single push operation.

To quantify the cost of the push operation, we set up a simulated network with 40 nodes. A push operation results in a broadcast transmission, which reaches all nodes in range. To create a situation in which the push operation was as expensive as possible, we set up our network so that all nodes are transmission range of each other. The nodes issue a push every ten seconds and we vary the amount of nodes that issue a push from one to all nodes. We measure the radio duty cycle of the nodes over the ten seconds between each push operation. The result is shown in the left graph in Fig. 10. As expected, we see that the cost grows linearly with the amount of nodes issuing a push.

The right graph in Fig. 10 shows the result of the same experiment, but with nodes issuing pull operations instead of push operations. We see that the cost is higher than for the push operation, but that it is relatively constant regardless of the number of nodes that issue a pull operation. This is due to the delay between the reception of a pull request and the corresponding push response: with many pull requests, nodes will receive several requests before eventually responding with a push. Thus the resulting power consumption is not significantly affected by the number of simultaneous pull operations.

5.3 Case Study: Collection and Dissemination

To study the aggregate effects of announcements on a real-world scenario, we perform a data collection testbed experiment. We use the Contiki shell to collect sensor data from a 24 node office testbed. The Contiki shell has one command for setting up a sink node, `collect`, which forms a collection tree with the Contiki collect protocol, and one command for sending data through the collection tree, `send`. To start the commands on the nodes in the network, the Contiki shell provides a mechanism for starting commands on other nodes in the network, `netcmd`. The `netcmd` command uses reliable data dissemination with Trickle to disseminate the commands through the network. Both the data collection protocol and the data dissemination protocols use beacons and we expect to see a reduction in the number of beacons in the network.

We run two versions of the experiment, one with announcement-based implementations of the protocols and one without. In both experiments we use ContikiMAC with a channel check rate of 8 Hz. We run the network for one hour for each experiment. With both experiments, we receive an average of 54 packets per node. Two nodes have poor connectivity and only reported 1 and 3 packets respectively in one experiment, and 1 and 7 packets in the other experiment, whereas the others reported 60 packets. The longest path was 5 hops long.

We measure the power consumption per Rime channel using Contiki's power profiler [6]. We see the resulting breakdown in Fig. 11. The boxes show the amount of transmission and reception power spent on beacons and data packets, respectively. The results show that the announcement-based implementation is

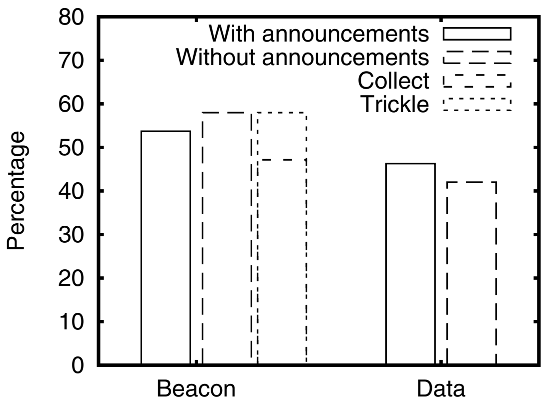


Fig. 11. Activity breakdown of the data collection and dissemination testbed experiment

able to reduce the number of beacons. The reduction is due to the suppression of data dissemination beacons, which account for 9% of the total number of beacons in the non-announcement-based implementation.

6 Related Work

The idea of inserting a new layer in the network stack to coordinate data from multiple upper-layer protocols has been used in many contexts. Balakrishnan et al. [1] introduced an explicit congestion management layer for Internet hosts. Choi et al. [3] add an isolation layer that shields different sensor network protocols from each other. The announcement layer is different because it focuses on a specific traffic type: broadcast beacons. Furthermore, since the announcement layer do not shield protocols from each other, there is no performance penalty as for the isolation layer by Choi et al [3].

There are many examples where information from multiple packets are combined into a single transmission to improve performance. Lin and Levis [16] observe that packing multiple pieces of information into the same physical packet aids in reducing the performance penalty due to broadcast transmissions. However, their scope is limited to information belonging to a single protocol (DIP), and comes hardwired with the protocol implementation itself. By contrast, the announcement layer provide a re-usable, generic mechanism that can be used across different protocols.

The push and pull operations of the announcement layer are similar to the operations used in sensor network neighborhood abstractions [18,22]. However, the latter aim at redefining the notion of physical neighborhood mostly based on application-level requirements. Announcements, instead, target network-level functionality that typically leverage communication in the physical neighborhood. In addition, some of the aforementioned systems [18,22] inherently provide a push-only communication paradigm, whereas announcements also provide a pull operation.

7 Conclusions

We present the announcement layer that piggybacks announcements from multiple protocols and coordinates their transmission to reduce the total amount of beacons. The background to the announcement layer is the observation that beacon transmissions are costly, and multiple transmissions even more so. In addition to beacon coordination, the announcement layer provides inter-protocol coordination through two operations: push and pull. We have implemented an announcement layer in Contiki and rewritten three staple sensornet protocols on top of it: data collection, single-packet data dissemination, and multi-packet data dissemination. We demonstrate that beacon coordination reduces the amount of beacons and that the cost of the push and pull operations is low.

Acknowledgments. This work was funded by the SSF, the Swedish Foundation for Strategic Research through the Promos and Supple projects, and VINNOVA, the Swedish Agency for Innovation Systems through the ReSense project.

References

1. Balakrishnan, H., Rahul, H., Seshan, S.: An integrated congestion management architecture for internet hosts. SIGCOMM Comput. Commun. Rev. 29(4), 175–187 (1999)
2. 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 the International Conference on Embedded Networked Sensor Systems (ACM SenSys), Boulder, Colorado, USA, pp. 307–320 (2006)
3. Choi, J., Kazandjieva, M., Jain, M., Levis, P.: The case for a network protocol isolation layer. In: Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys), Berkeley, CA, USA (2009)
4. Costa, P., Mottola, L., Murphy, A.L., Picco, G.P.: Programming wireless sensor networks with the TeenyLIME middleware. In: Cerqueira, R., Pasquale, F. (eds.) Middleware 2007. LNCS, vol. 4834, pp. 429–449. Springer, Heidelberg (2007)
5. Deshpande, A., Guestrin, C., Madden, S.: Resource-aware wireless sensor-actuator networks. IEEE Data Engineering 28(1) (2005)
6. Dunkels, A., Österlind, F., Tsiftes, N., He, Z.: Software-based on-line energy estimation for sensor nodes. In: Proceedings of the IEEE Workshop on Embedded Networked Sensor Systems (IEEE Emnets), Cork, Ireland (June 2007)
7. Dunkels, A., Österlind, F., He, Z.: An adaptive communication architecture for wireless sensor networks. In: Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys), Sydney, Australia (November 2007)
8. Dutta, P., Culler, D.: Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In: Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys), Raleigh, North Carolina, USA (2008)
9. 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)
10. El-Hoiydi, A., Decotignie, J.D., Enz, C.C., Roux, E.L.: Wisemac, an ultra low power mac protocol for the wisenet wireless sensor network. In: Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys), pp. 302–303 (2003)
11. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection tree protocol. In: Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys), Berkeley, CA, USA (2009)
12. Hui, J.W., Culler, D.: The dynamic behavior of a data dissemination protocol for network programming at scale. In: Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys), Baltimore, Maryland, USA (November 2004)
13. Jiang, J., Tseng, Y., Hsu, C., Lai, T.: Quorum-based asynchronous power-saving protocols for IEEE 802.11 ad hoc networks. Mobile Networks Applications 10(1-2) (2005)

14. Kandhalu, A., Lakshmanan, K., Rajkumar, R.: U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol. In: Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN), Stockholm, Sweden, pp. 350–361 (2010)
15. Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In: Proceedings of the USENIX Symposium on Networked Systems Design & Implementation (NSDI 2004) (March 2004)
16. Lin, K., Levis, P.: Data discovery and dissemination with dip. In: Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN), St. Louis, MO, USA (2008)
17. Moss, D., Levis, P.: BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking. Tech. Rep. SING-08-00, Stanford University (2008)
18. Mottola, L., Picco, G.: Logical neighborhoods: A programming abstraction for wireless sensor networks. In: IEEE International Conference on Distributed Computing in Sensor Systems (2006)
19. Polastre, J., Hill, J., Culler, D.: Versatile low power media access for wireless sensor networks. In: Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys), pp. 95–107. ACM Press, Baltimore (2004)
20. Tsiftes, N., Eriksson, J., Finne, N., Österlind, F., Höglund, J., Dunkels, A.: A Framework for Low-Power IPv6 Routing Simulation, Experimentation, and Evaluation. In: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM), Demo Session, New Delhi, India (August 2010)
21. Vasseur, J., Dunkels, A.: Interconnecting Smart Objects with IP: The Next Internet. Morgan Kaufmann, San Francisco (2010)
22. Whitehouse, K., Sharp, C., Brewer, E., Culler, D.: Hood: a neighborhood abstraction for sensor networks. In: Proceedings of The International Conference on Mobile Systems, Applications, and Services (MobiSys), Boston, MA, USA (June 2004)
23. Winter, T., Thubert, P.(eds.) : RPL Author Team: RPL: IPv6 Routing Protocol for Low power and Lossy Networks, internet Draft draft-ietf-roll-rpl-11 (work in progress)
24. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges of reliable multihop routing in sensor networks. In: Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys), pp. 14–27. ACM Press, Los Angeles (2003), `http://citeseer.ist.psu.edu/woo03taming.html`
25. Zheng, R., Hou, J., Sha, L.: Asynchronous wakeup for ad hoc networks. In: Proceedings of the 4th ACM International Symposium on Mobile ad hoc Networking & Computing (2003)

Author Index