Piet Demeester
Ingrid Moerman
Andreas Terzis (Eds.)

LNCS 7772

Wireless Sensor Networks

10th European Conference, EWSN 2013
Ghent, Belgium, February 2013
Proceedings

Springer

Lecture Notes in Computer Science 7772

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

Piet Demeester Ingrid Moerman
Andreas Terzis (Eds.)

Wireless Sensor Networks

10th European Conference, EWSN 2013
Ghent, Belgium, February 13-15, 2013
Proceedings

Springer

Volume Editors

Piet Demeester
Ingrid Moerman
Ghent University - iMinds
Department of Information Technology (INTEC)
Gaston Crommenlaan 8 Box 201, 9050 Ghent, Belgium
E-mail: {piet.demeester, ingrid.moerman}@intec.ugent.be

Andreas Terzis
Johns Hopkins University
Computer Science Department
204 Shaffer Hall, 3400 North Charles Street, Baltimore, MD 21218, USA
E-mail: terzis@cs.jhu.edu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-36671-0 e-ISBN 978-3-642-36672-7
DOI 10.1007/978-3-642-36672-7
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013931225

CR Subject Classification (1998): C.2.0-4, C.3, C.4, E.4, F.2.2, H.2.m

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 2013, the 10th European Conference on Wireless Sensor Networks. The conference took place in Ghent, Belgium, during February 13–15, 2013. 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 51 paper submissions of which 12 were selected for publication and presentation, yielding an acceptance rate of 23 %. Paper submissions were received from 24 different countries from all over 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 224 reviews with all papers being evaluated by at least three independent reviewers, and most receiving 4 to 6 reviews. In addition, the Technical Program Committee had an online discussion that went through the details of every paper in isolation and in comparison with the other submitted papers, prior to making final decisions. All accepted papers underwent an additional round of revision managed by a "shepherd," appointed among the reviewers. The final program covered a wide range of topics, which were grouped into four sessions: Experimentation and Data Access; Data Management; Network Algorithms and Protocols; and Physical Layer and Hardware Aspects.

The conference program included other elements in addition to the presentation of research papers. The keynote was given by JP Vasseur, Cisco Fellow, and one of the drivers behind the Internet of Things who spoke about "The Internet of Things and IETF standardization." A poster and research demo session, co-chaired by Omprakash Gnawali (University of Houston, USA) and Chris Blondia (University of Antwerp - iMinds, Belgium), attracted 33 submissions, for which separate proceedings are available. Two tutorials were also held before the conference: "Building Reconfigurable Wireless Sensor Network Applications on Heterogeneous Infrastructure" by Danny Hughes and David Cuartielles (iMinds-DistriNet, KU Leuven) and "A Hands-On Introduction to Running Large-Scale Wireless Sensor Networking Experiments on the iMinds w-iLab.t Testbed" by Stefan Bouckaert (iMinds). Finally, for the first time this year, besides several industry demonstrators, a special session was also held on "The Future of the Sensor Industry" with the aim of educating sensor researchers about the "do's and don'ts" when setting up a (start-up) sensor company.

We would like to thank everyone who contributed to EWSN 2013. In particular, we would like to thank the Technical Program Committee for their reviews, and the entire Organizing Committee for their support. Finally, we also would like to thank the local administration at the IBCN research group from Ghent

University for their help with the conference planning, and last, but certainly not least, we thank our sponsors: iMinds, Ghent University, and Interuniversity Attraction Poles Programme BESTCOM initiated by the Belgian Science Policy Office.

February 2013

Ingrid Moerman
Andreas Terzis

Organization

General Chair

Piet Demeester	Ghent University - iMinds, Belgium

Program Chairs

Ingrid Moerman	Ghent University - iMinds, Belgium
Andreas Terzis	Johns Hopkins University, USA

Local Organization

Eli De Poorter	Ghent University - iMinds, Belgium

Poster Chair

Omprakash Gnawali	University of Houston, USA

Demo Chair

Chris Blondia	University of Antwerp - iMinds, Belgium

Publicity Chair

Luca Mottola	Politecnico di Milano, Italy, and Swedish Institute of Computer Science, Sweden
Marcus Chang	Johns Hopkins University, USA
Peter Van Daele	Ghent University - iMinds, Belgium

Webmaster

Jeroen Hoebeke	Ghent University - iMinds, Belgium

Technical Program Committee

Akos Ledeczi	Vanderbilt University, USA
Amy L. Murphy	Bruno Kessler Foundation - CIT IRST, Italy
Anna Forster	SUPSI, Switzerland
Bhaskar Krishnamachari	USC, USA
Brano Kusy	CSIRO, Australia
Cecilia Mascolo	University of Cambridge, UK
Cem Ersoy	Bogazici University, Turkey
Chiara Petrioli	University of Rome "La Sapienza", Italy
Cormac J. Sreenan	University College Cork, Ireland
Eli De Poorter	Ghent University - iMinds, Belgium
Emiliano Miluzzo	AT&T labs, USA
Xiaofan Jiang	Intel Labs, China
Geoffrey Challen	University at Buffalo, USA
Gian Pietro Picco	University of Trento, Italy
Holger Karl	University of Paderborn, Germany
Ilker Demirkol	Universitat Politecnica de Catalunya, Spain
Jan Beutel	ETH Zurich, Switzerland
Jie Gao	Stony Brook University, USA
Kamin Whitehouse	University of Virginia, USA
Katia Jaffres-Runser	University of Toulouse/IRIT Laboratory/ INPT ENSEEIHT, France
Kay Roemer	University of Lübeck, Germany
Kirk Martinez	University of Southampton, UK
Klaus Wehrle	RWTH Aachen University, Germany
Koen Langendoen	TU Delft, The Netherlands
Krishna Sivalingam	Institute of Technology Madras, India
Leo Selavo	University of Latvia, Latvia
Luca Mottola	Politecnico di Milano, Italy, and Swedish Institute of Computer Science, Sweden
Mario Alves	Politecnico do Porto (ISEP/IPP), CISTER Research Unit, Portugal
Michele Zorzi	University of Padova, Italy
Mingyan Liu	University of Michigan, USA
Nael Abu-Ghazelah	Binghamton University, USA
Pedro Jose Marron	University of Duisburg-Essen and Fraunhofer FKIE, Germany
Philippe Bonnet	Copenhagen University, Denmark
Rajeev Shorey	NIIT University, India
Salil Kanhere	The University of New South Wales, Australia
Sam Michiels	KULeuven, Belgium
Tarek Abdelzaher	UIUC, USA
Tommaso Melodia	State University of New York at Buffalo, USA
Violet R. Syrotiuk	Arizona State University, USA
Wendi Heinzelman	University of Rochester, USA

Table of Contents

On Assessing the Accuracy of Positioning Systems in Indoor Environments

Hongkai Wen, Zhuoling Xiao, Niki Trigoni, and Phil Blunsom

University of Oxford
{name.lastname}@cs.ox.ac.uk

Abstract. As industrial and academic communities become increasingly interested in Indoor Positioning Systems (IPSs), a plethora of technologies are gaining maturity and competing for adoption in the global smartphone market. In the near future, we expect busy places, such as schools, airports, hospitals and large businesses, to be outfitted with multiple IPS infrastructures, which need to coexist, collaborate and / or compete for users. In this paper, we examine the novel problem of estimating the accuracy of co-located positioning systems, and selecting which one to use where. This is challenging because 1) we do not possess knowledge of the ground truth, which makes it difficult to empirically estimate the accuracy of an indoor positioning system; and 2) the accuracy reported by a positioning system is not always a faithful representation of the real accuracy. In order to address these challenges, we model the process of a user moving in an indoor environment as a Hidden Markov Model (HMM), and augment the model to take into account vector (instead of scalar) observations, and prior knowledge about user mobility drawn from personal electronic calendars. We then propose an extension of the Baum-Welch algorithm to learn the parameters of the augmented HMM. The proposed HMM-based approach to learning the accuracy of indoor positioning systems is validated and tested against competing approaches in several real-world indoor settings.

1 Introduction

We are only on the cusp of the wave of Indoor Positioning Systems (IPS), but there is already an explosion of technologies that compete for adoption in the global smart device market. These range from ultrasound-based positioning systems, to Bluetooth Low Energy, RFID systems, WiFi-based localization (with or without fingerprinting), inertial tracking, ground-based transmitters to extend GPS service indoors, visible light comms, and sensing ambient FM-radio, magnetic, and photo-acoustic signatures.

The above list is not exhaustive; the interest and competition in this area is growing so fast that an alliance of 22 industries, called In-Location, was founded in August 2012 to promote the adoption and deployment of IPS solutions. In the current climate, it is reasonable to expect that many indoor environments, such as schools, airports, hospitals and large businesses, will soon be outfitted with multiple IPS (Indoor Positioning System) infrastructures, which need to coexist, collaborate and / or compete for users. In this emerging ecosystem, it is important for users to be able to *accurately estimate the accuracy of an IPS at different locations of an indoor environment*. We first define

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 1–17, 2013.

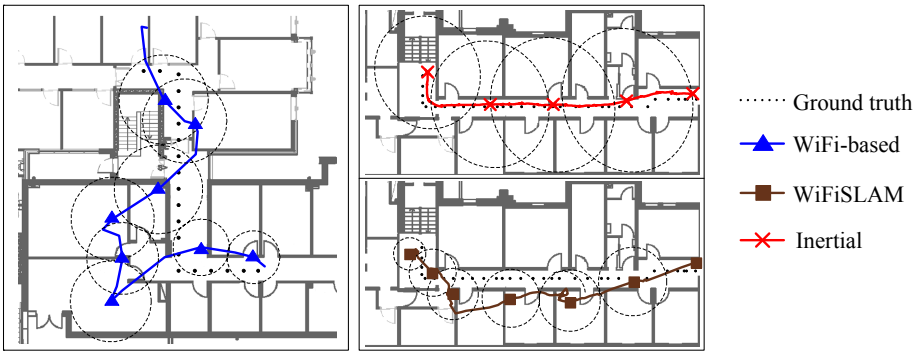


Fig. 1. (a) At the bottom left corner, the true locations fall out of the reported error ellipses. (b) The reported accuracies are misleading as to which is the best IPS.

what we mean by IPS *accuracy*, and then explain why it is both useful and challenging to estimate it.

IPS Accuracy: Consider the simple case where an IPS provides only location estimates without uncertainty information associated with them. Assume that the true location of the user at a given time is x and the measurement generated by the IPS is z. The accuracy ϵ of the IPS at location x depends on the Euclidean distance between the true and measured locations: $\epsilon(x) = f(\|z - x\|_2)$, where f is a monotonic increasing function. Many IPSs tend to also provide accuracy information, for example, they pair a location estimate with an error ellipse around it and a confidence interval. In such cases, the measurement z of an IPS can be viewed as a random variable with a probabilistic distribution function $\mathrm{P}_z : L \rightarrow [0, 1]$, where L is the sample space over all possible locations. With respect to the real location x, the accuracy of an uncertain measurement z depends on the *expected* Eculidean distance between the true and measured locations: $\epsilon(x) = f(\sum_{\ell \in L} \mathrm{P}_z(\ell)(\|\ell - x\|_2))$.

Why Is It Useful to Estimate IPS Accuracy: Users will be able to make informed decisions as to which IPS to task when moving in an indoor space, and when to switch from one IPS to another. Of course, accuracy is not the only information required to make such decisions. Cost, power consumption and privacy should also be taken into consideration. It is possible that highly accurate IPSs may impose a small monetary cost on users whereas less accurate ones may be available for free. Also, the power consumption of different IPSs can vary up to magnitudes, and some users may be more concerned about privacy than others. In this paper, we focus on estimating accuracy, and assume that other quality metrics are available from other sources (energy monitoring tools, web sources, etc.). We claim that by providing accuracy information, we empower users to choose the IPS that achieves the desired tradeoff between accuracy and the remaining quality metrics.

Why is it Difficult to Estimate IPS Accuracy: First, the actual trajectory of a user is normally unavailable, and thus the accuracy of an IPS can not be directly computed.

Second, the accuracy of an IPS can vary significantly over space. Hence, we cannot rely on the accuracy of an IPS in one location to infer its accuracy in another. Thirdly, we view an IPS as a black box, and assume little or no knowledge about how it works or where it has been deployed.

Finally, the reported accuracy from an IPS (if provided) is not always a reliable indicator of the real accuracy. We *observe two general cases where this happens, and* provide illustrative examples: *Case 1:* The true location of a user consistently falls out of the reported ellipsoid. This tends to happen when the localization algorithm of an IPS can not cope with changes in the environment. For instance in Fig. 1(a), an IPS that uses WiFi RSSI measurements performs well in the straight corridor, but consistently underestimates the error in its measurements when the node is moving towards the corner. Another example concerns safety-critical environments, where one or more IPSs may be compromised and made to consistently report wrong locations with very high reported accuracy. *Case 2:* The size of ellipsoids provided by different IPSs can lead us to wrong conclusions about which one is more accurate. For instance, consider the two estimated trajectories of a single user shown in Fig. 1(b), where the top one is provided by an IPS using inertial sensors and the other is from WiFiSLAM [5]. One can immediately see that the estimated trajectory on top is far better than the bottom one, but the reported accuracy is significantly lower[1]. The aim of this paper is to address these challenges, and provide a solution to the IPS accuracy estimation problem.

Summary of Contributions: In this paper, we make the following four contributions:

1. To our knowledge, this is the first paper that defines the problem of assessing the accuracy of co-located indoor positioning systems. As explained above, this is a *timely* problem, and an effective solution could benefit a global market of IPS users.
2. We propose a maximum likelihood HMM-based approach to assessing IPS accuracy. We model the stochastic process of a node moving through an area as a HMM (Hidden Markov Model), which we augment: 1) to accommodate non-scalar probabilistic observations, and 2) to incorporate prior information on the user trajectory drawn from personal calendars. We cast the problem of IPS accuracy estimation to that of learning the parameters of the augmented HMM.
3. We propose a novel Expectation Maximization algorithm to solve the HMM learning problem. This algorithm extends the classic Baum-Welch algorithm [18] to account for non-scalar observations and calendar constraints. *We then show how the learnt HMM parameters can be used to evaluate the accuracy of co-located IPSs.*
4. We evaluate the performance of the learning algorithm in a variety of indoor environments. We show that in most cases the learnt accuracy of an IPS is far more accurate than the reported one. Finally, we demonstrate the first *IPS recommendation system*, which takes into account IPS accuracy, and recommends which IPS system to use where.

The rest of the paper is organized as follows: Section 5 overviews related work. Section 2 casts the problem into that of learning the parameters of a HMM. Section 3 explains how to solve the problem by extending the classic Baum-Welch algorithm.

[1] Note that in both cases the real trajectory is covered by the reported ellipses.

Section 4 evaluates the proposed and competing approaches in various indoor environments. Finally, Section 6 concludes the paper, and discusses ideas for future work.

2 Problem Formulation

In this section, we model the movement of a user in an indoor environment as a HMM[13], which we augment to account for probabilistic observations. We show how to incorporate into the model prior information about a user's schedule found in her calendar. We then show how the problem of assessing the accuracy of IPSs can be cast into that of estimating the parameters of the augmented HMM.

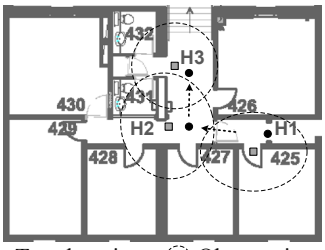
2.1 Definition of HMM Parameters

Suppose that M Indoor Positioning Systems (IPSs) are co-located in an indoor environment and provide positioning services to a user. Given the map of the environment, space is preprocessed into a finite set of N locations $L = \{l_k\}$, $1 \leq k \leq N$, e.g. different rooms or corridor segments in a building. In our experiments, the average size of the discrete locations is 3m×3m. At any discrete timestamp t, $1 \leq t \leq T$, an IPS reports an observation z_t of the user's true location x_t, which is typically unknown.

Hidden States: We model the true locations of the user as the hidden states. We denote the event that the true location of the user at time t is l_j as $x_t = l_j$, where $l_j \in L$. We also implicitly assume the actual trajectory $[x_1, ..., x_T]$ of a user is *Markovian*, i.e. $\mathrm{P}(x_t|x_1, ..., x_{t-1}) = \mathrm{P}(x_t|x_{t-1})$.

State Transition Probabilities and Initial State Distribution: The probability of a user moving from one location to another is governed by the transition probability distribution $A = \{a_{ij}\}$, where $a_{ij} = \mathrm{P}(x_{t+1} = l_j|x_t = l_i)$, $1 \leq i, j \leq N$. In an indoor environment, the transition probabilities a_{ij} largely depend on the underlying map. We assume that transitions can only be made from a given location to the same or adjacent locations with equal probability. Finally, the initial state distribution defines the probabilities that a user starts at each of the N possible locations: $\pi_i = \mathrm{P}(x_1 = l_i)$, $1 \leq i \leq N$, which is set to uniform in our experiments.

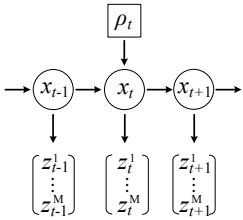
Observations: Unlike a standard HMM, which emits a single symbol per observation, in our case each observation is a *probability distribution*. For ease of exposition, let's initially assume that we have one IPS (referred to as the *system*). Under the assumption of discrete space, an observation z_t reported by the system at time t is a multinomial distribution $z_t = [p_t(1), ..., p_t(N)]$, where $p_t(k)$, $1 \leq k \leq N$, is the belief of the system that the user is in location l_k at time t. For example, the probabilistic observations (ellipses) of the system in Fig. 2(a) can be represented as vectors of probabilities shown in Fig. 2(b), where the tuple $\langle \mathrm{H2}, 0.4 \rangle$ at time t indicates that the system places the user at the corridor H2 with 0.4 probability. In the presence of M IPSs, the observation at time t becomes an $M \times N$ matrix $\{z_t^m\}$, where the m-th row is a vector of probabilities $[p_t^m(1), ..., p_t^m(N)]$ reported by the m-th IPS. For simplicity we assume there is no environment dynamics or external interferences that may affect the co-located IPSs,



(a)

	$t-1$		t		$t+1$
425	0.3	425	0	425	0
426	0.05	426	0	426	0
427	0.15	427	0.2	427	0
428	0	428	0.1	428	0
429	0	429	0	429	0
430	0	430	0	430	0
431	0	431	0.1	431	0.1
432	0	432	0	432	0.1
H1	0.5	H1	0	H1	0
H2	0	H2	0.4	H2	0.2
H3	0	H3	0.2	H3	0.6

(b)



(c)

Fig. 2. (a) Probabilistic observations (ellipses) at time $t-1$, t, and $t+1$. (b) Probabilistic observations can be represented as vectors of probabilities. (c) The augmented HMM.

and thus observations from different IPSs are conditionally independent given the real trajectory of the user, i.e. $\mathrm{P}(z_t^m, z_t^{m'}|x_t) = \mathrm{P}(z_t^m|x_t)\mathrm{P}(z_t^{m'}|x_t)$, for $1 \leq m, m' \leq M$.

Symbol Emission Probabilities: We now explain the symbol emission probabilities of the augmented HMM that admits probabilistic observations. We first assume a single IPS for simplicity. Let $B = \{b_j(k)\}$ be the stochastic matrix of emission probabilities. Each element $b_j(k)$ is defined as the average probability that the system assigns to location l_k, over all possible observations made when the user is actually in location l_j, $1 \leq j, l \leq N$. This probability distribution is different from the emission probabilities $B_{std} = \{b_j(k)_{std}\}$ in standard HMMs, where $b_j(k)_{std}$ is the probability that the user is observed in l_k when the actual location is l_j, $1 \leq j, k \leq N$. In standard HMM, likelihood of an observation at t given the state depends only on B_{std}, because z_t is *scalar* (a single location), and the sample space of z_t is the finite set L. In our model, the probability of observing z_t given the state can not be fully captured by B since z_t becomes a *vector* of symbols associated with different probabilities, which can not be modeled by any finite distributions.

When M IPSs coexist, under the assumption that the observations from different IPSs are independent conditioned on the states, the symbol emission probabilities for our model is a collection of M stochastic matrices $\boldsymbol{B} = \{B_m\}$, where each B_m is the individual symbol emission distribution for the m-th IPS.

In summary, our model $\lambda = (A, \boldsymbol{B}, \pi)$ is a generalization of standard HMMs, which supports probabilistic observations, and considers observations from multiple IPSs.

2.2 Accuracy Assessment as Parameter Estimation

Now we show how the problem of assessing the accuracy of co-located IPSs at different locations can be cast into that of learning the emission probabilities of the HMM λ defined above. Recall from Section 1 that the accuracy ϵ of an IPS at location x is defined as a function of the expected distance between the true and measured locations. Also, recall that the symbol emission probability $b_j(k)$ of an IPS represents the average probability associated with location l_k in the observations when the real state is $x = l_j$. We thus have:

$$\epsilon(l_j) = f(\sum_{l_k \in L} \mathrm{P}(l_k \text{ is reported in } z | x = l_j) \|l_k - l_j\|_2) = f(\sum_{l_k \in L} b_j(k) \|l_k - l_j\|_2) \quad (1)$$

where $\|l_k - l_j\|_2$ is the Euclidean distance between locations l_j and l_k. More generally, in a multi-IPS environment, the accuracy of the m-th IPS at different locations is governed by its symbol emission probabilities B_m. Since the actual user trajectories are unknown, we cannot empirically compute each B_m. The only information we can rely on is the observations Z of the M coexisting IPSs. Therefore, our goal is to find the model parameters λ that are most consistent with the observed data Z. Formally, this is given by the maximum likelihood estimate $\hat{\lambda}_{mle} = \arg\max_\lambda \mathrm{P}(Z|\lambda)$.

2.3 Incorporating User Provided Prior Knowledge

In this section, we explore the idea of using private information found in users' calendars, as clues on the user locations. We refer to such information as *priors*, and show how to incorporate them in our model. Formally, at a given timestamp t, a prior ρ_t is a distribution $\rho_t = [p_t(1), ..., p_t(N)]$ over the N discrete locations L, where each $p_t(k)$, $1 \leq k \leq N$, is the probability that the user is in location l_k at time t. The priors typically come from planned events, such as doctor appointments, flight boarding times and work meetings. Techniques of parsing the knowledge of such calendar information into priors are beyond the scope of this paper, and in our experiments, we adopt a simple approach that assigns a large probability at the location where the event is supposed to take place. For instance, a calendar entry of "meeting in room 100 at 3pm" can be interpreted as a prior $\rho_{3\text{pm}}$, with 0.9 probability associated with room 100, and the rest of the probability mass uniformly distributed elsewhere. We incorporate such knowledge as a prior distribution of the hidden states at time t, as shown in Fig. 2(c). Since we have extra information about the world, the optimization goal becomes finding the best model parameters $\hat{\lambda}_{mle} = \arg\max_\lambda \mathrm{P}(Z, \boldsymbol{\rho}|\lambda)$, which maximizes the joint likelihood of both the observations Z and the set of observed priors $\boldsymbol{\rho}$.

3 Proposed Algorithm

In this section, we first solve the parameter estimation problem formulated in the previous section, and then describe the complete algorithm that estimates the accuracy of IPSs. Recall from the previous section that our goal is to find the optimal model parameters $\lambda = (A, \boldsymbol{B}, \pi)$ that can best explain the observed data, which in our case, are the observations from co-located IPSs and the user-provided priors. The proposed algorithm extends the classic Baum-Welch Expectation Maximization algorithm to account for probabilistic observations and to exploit prior information drawn from users' calendars. We first describe how to derive the likelihood of the observed data with our model, and then show how to find the model parameters that maximize the likelihood.

3.1 Derive the Likelihood of Data

Given the observed data $O = \{Z, \boldsymbol{\rho}\}$, where Z are the observations from the M coexisting positioning systems and $\boldsymbol{\rho}$ the calendar priors provided by the user, the likelihood

of data can be computed by summing out all possible hidden state sequences. We assume that given a hidden state sequence X (the true user trajectory), the priors $\boldsymbol{\rho}$ and observations Z from IPSs are conditionally independent. We also assume observation sequences from one IPS Z_m, $1 \leq m \leq M$, are conditionally independent of observations from the other IPSs given X. Then the likelihood of data is:

$$L_\lambda(O) = \sum_{X \in \mathcal{X}} \mathrm{P}(O, X|\lambda) = \sum_{X \in \mathcal{X}} \mathrm{P}(\boldsymbol{\rho}, X|\lambda) \prod_{m=1}^{M} \mathrm{P}(Z_m|X, \lambda) \tag{2}$$

Let us now explain how we derive components $\mathrm{P}(\boldsymbol{\rho}, X|\lambda)$ and $\mathrm{P}(Z_m|X, \lambda)$, starting with the first one that concerns the set of priors $\boldsymbol{\rho}$ provided by the user. Suppose that a prior ρ_t is given at timestamp t. ρ_t is a distribution $[q_t(1), ..., q_t(N)]$ over the set of locations L, where each $q_t(k)$ is the probability that the user is in location l_k, $1 \leq k \leq N$. If we assume the priors are trustworthy, then ρ_t essentially makes the hidden state visible, i.e. the distribution of the state x_t is known. In essence, the set of priors $\boldsymbol{\rho}$ *constrains* the dynamics of the model. For instance consider an extreme example where a prior ρ_t places probability 1 at a single location l_k, and 0 elsewhere. In this case from time $t-1$ to t, there will be no state transition to any locations other than l_k, and the state of our model at t is restricted to l_k only. Assume for simplicity that there is no prior at the beginning of time $t = 1$. Given a fixed state sequence $X = \{x_1, ...x_T\}$, where x_t is the true location of the user at t, the joint probability of the priors $\boldsymbol{\rho}$ and the state sequence X conditioned on the model λ is:

$$\mathrm{P}(\boldsymbol{\rho}, X|\lambda) = \mathrm{P}(x_1|\lambda) \prod_{t=2}^{T} \mathrm{P}(x_t|x_{t-1}) = \pi_{x_1} \prod_{t=2}^{T} \tilde{a}_{x_{t-1}x_t}(t) \tag{3}$$

where $\tilde{a}_{ij}(t)$ is the probability of transiting from state i at time $t-1$ to state j at time t under the constraints specified by the prior:

$$\tilde{a}_{ij}(t) = \begin{cases} a_{ij}, & \text{if no prior is observed at } t \quad (4a) \\ q_t(j), & \text{if } \rho_t \text{ is observed at } t \quad (4b) \end{cases}$$

Let us now derive component $\mathrm{P}(Z_m|X, \lambda)$ of Eqn. 2. Let z_t^m be the observation of the m-th IPS at time t. Recall that z_t^m is a vector of probabilities $[p_t^m(1), ..., p_t^m(N)]$, where $p_t^m(k)$, $1 \leq k \leq N$, indicates the belief of the m-th positioning system that the user is in location l_k at time t. The probability of the observation z_t^m given the state x_t and model λ, can be computed by summing over all possible locations l_k as follows:

$$\mathrm{P}(z_t^m|x_t, \lambda) = \sum_{k=1}^{N} p_t^m(k) \mathrm{P}(z_t^m = l_k|x_t, \lambda) = \sum_{k=1}^{N} p_t^m(k) b_{x_t}^m(k) \tag{5}$$

where recall that $b_{x_t}^m(k)$ denotes the emission probability of location l_k at state x_t by the m-th IPS. Then the probability of observation sequence Z_m given a fixed state sequence X and the model λ is the product of the observation likelihood at each timestamp:

$$\mathrm{P}(Z_m|X, \lambda) = \prod_{t=1}^{T} \sum_{k=1}^{N} p_t^m(k) b_{x_t}^m(k)) \tag{6}$$

3.2 Maximize the Likelihood of Data

Our goal is to find model parameter $\lambda = (A, \boldsymbol{B}, \pi)$ that maximizes the likelihood of observed data $L_\lambda(O)$. This can be achieved by the general EM algorithm, and in our case, the idea is to iteratively perform the following two steps of computation: (1) for the current model parameter λ, derive the expected log likelihood function $Q(\lambda', \lambda)$ over a new parameter λ'; and (2) find the λ' that maximizes the Q function, and in the next iteration substitute λ with this newly estimated λ'.

Now we show how to derive the new parameter $\lambda' = (A', \boldsymbol{B}', \pi')$ with the above procedure. Using Eqn. (2), the Q function can be represented as follows:

$$Q(\lambda', \lambda) = \sum_{X \in \mathcal{X}} \mathrm{P}(O, X|\lambda) \Big[\log(\mathrm{P}(\boldsymbol{\rho}, X|\lambda')) + \sum_{m=1}^{M} \log(\mathrm{P}(Z_m|X, \lambda')) \Big] \tag{7}$$

where the terms $\log(\mathrm{P}(\boldsymbol{\rho}, X|\lambda')$ and $\log(\mathrm{P}(Z_m|X, \lambda')$ can be further expanded using Eqns. (3) and (6). To get the parameter $\lambda' = (A', \boldsymbol{B}', \pi')$ that maximizes Q, we use the Lagrange multiplier method, and set the partial derivatives of the resulting objective function to zero, which yields:

$$\begin{aligned}
&\pi'_i = \frac{\alpha_1(i)\beta_1(i)}{\sum_{1}^{N} \alpha_1(i)\beta_1(i)}; \quad a'_{ij} = \frac{\sum_{2}^{T} \Big[\alpha_{t-1}(i)\tilde{a}_{ij}(t) \Big(\prod_{m=1}^{M} \sum_{k=1}^{N} p_t^m(k) b_j^m(k) \Big) \beta_t(j) \Big]}{\sum_{2}^{T} \alpha_{t-1}(i)\beta_{t-1}(i)}; \\
&b_j^m(k)' = \frac{1}{\sum_{1}^{T} \alpha_t(j)\beta_t(j)} \Big\{ \pi_j \Big[p_1^m(k) b_j^m(k) \prod_{m' \neq m}^{M} \sum_{k=1}^{N} p_1^{m'}(k) b_j^{m'}(k) \Big] + \\
&\sum_{2}^{T} \Big[\sum_{i=1}^{N} \alpha_{t-1}(i)\tilde{a}_{ij}(t) \Big] \Big[p_1^m(k) b_j^m(k) \prod_{m' \neq m}^{M} \sum_{k=1}^{N} p_1^{m'}(k) b_j^{m'}(k) \Big] \beta_t(j) \Big\}
\end{aligned} \tag{8}$$

where the variables α and β in the right hand sides of the above are *forward* and *backward* variables, and the definitions of these variables are extended from the original Baum-Welch algorithm to account for probabilistic observations and to exploit prior information about calendar constraints.

Forward Variables: We define the forward variable $\alpha_t(j)$ as the joint probability of having the probabilistic observations and priors in the first t steps, and of landing at state l_j at time t given the model:

$$\alpha_t(j) = \mathrm{P}(\rho_{1:t}, \boldsymbol{z}_{1:t}, x_t = l_j|\lambda) = \Big[\sum_{i=1}^{N} \alpha_{t-1}(i)\tilde{a}_{ij}(t) \Big] \prod_{m=1}^{M} \sum_{k=1}^{N} p_t^m(k) b_j^m(k) \tag{9}$$

where $b_j^m(k)$ is from the symbol emission probability distribution of the m-th IPS, $p_t^m(k)$ is the probability assigned to location l_k by the m-th IPS in its observation at time t, and $\tilde{a}_{ij}(t)$ is the probability of transiting from state l_i (at time $t-1$) to l_j (at time t) under the constraints specified by the prior (see Eqn. 4).

Algorithm 1. HMM Inference

1: Initialize $\alpha_1(i) \leftarrow \pi_i$
2: **for** $t = 2 : T$ **do**
3: Compute $\alpha_t(i)$ as in (9)
4: **end for**
5: Compute $L_\lambda(O) \leftarrow \sum_{i=1}^{N} \alpha_T(i)$
6: Initialize $\beta_T(i) \leftarrow 1$
7: **for** $t = T - 1 : 1$ **do**
8: Compute $\beta_t(i)$ as in (10)
9: **end for**

Algorithm 2. Accuracy Estimation

1: **while** $L_\lambda(O)$ does not converge **do**
2: Perform inference as in Algo. 1
3: Compute λ' as in (8) and do: $\lambda \leftarrow \lambda'$
4: **end while**
5: **for** each IPS ps_m **do**
6: **for** location $l_j \in L$ **do**
7: $\epsilon^m(l_j) \leftarrow \sum_{k=1}^{N} b_j^m(k) \|l_k - l_j\|_2$
8: **end for**
9: **end for**

Backward Variables: We define the backward variable $\beta_t(i)$ as the joint probability of observations and priors from time $t+1$ to T given state l_i at time t and the model λ:

$$\begin{aligned}\beta_t(i) &= \mathrm{P}(\rho_{t+1:T}, \boldsymbol{z}_{t+1:T} | x_t = l_i, \lambda) \\ &= \Big[\sum_{j=1}^{N} \tilde{a}_{ij}(t+1) \prod_{m=1}^{M} \sum_{k=1}^{N} p_{t+1}^m(k) b_j^m(k) \Big] \beta_{t+1}(j)\end{aligned} \tag{10}$$

The procedure of evaluating the forward and background variables are shown in Algo. 1, which is equivalent with the above inductive definitions. Now we are in the position of presenting the accuracy estimation algorithm.

3.3 The Accuracy Estimation Algorithm

As shown in Algo. 2, in each iteration the accuracy estimation algorithm first calls the inference algorithm (Algo. 1), which scans the observed data (observations from the IPSs and priors from the user) twice to compute the forward and backward variables. Then it computes the new parameters of the model that maximize the likelihood of data, and uses the newly estimated parameters for the next iteration, until the likelihood of data converges. Then for each IPS, the algorithm estimates its accuracy at a location l_j by computing the expected distance between the true and measured locations, from the learnt symbol emission probability distributions.

From Eqn. 9 and Eqn. 10, we can see the complexity of HMM inference (Algo. 1) is on the order of $O(MN^3T)$, where M is the number of co-located IPSs, N is the amount of discrete locations, and T is the length of the observation sequences. The complexity of accuracy estimation algorithm (Algo. 2) is then $O(IMN^3T)$, where I is the number of iterations until converge. In practice M is typically small and the probabilistic observations $z_t^m = [p_t^m(1), ..., p_t^m(N)]$ from an IPS can be very sparse. Therefore with some optimizations our algorithm is not prohibitively expensive comparing to the classic Baum-Welch algorithm, which is of order $O(IN^2T)$.

4 Evaluation

4.1 Experiment Setup

We evaluate our proposed algorithm in two indoor settings. One is the 4th floor of a CS department, which is a typical office environment. The other is the basement of a building, which includes seminar rooms, lecture theaters and other common areas. Each space is outfitted with multiple co-located indoor positioning systems, as shown in Fig. 3. Four different WiFi-based positioning systems, ps_1 to ps_4, operate on the 4th floor, and three systems: ps_5 to ps_7 on the basement. Each IPS owns a set of WiFi basestations at different areas of the space. The basestations periodically broadcast WiFi beacons, which are then picked up by the mobile devices carried by the users. An IPS estimates location by combining RSSI measurements from its basestations with inertial data collected from a foot mounted IMU. We track two users carrying different mobile devices (a Nexus S and an Asus TF201 tablet) during their work time and collect data for 20 days on the 4th floor and 4 days in the basement. The ground truth is provided by the users: the floor plans are presented at their devices and they touch the actual positions they are visiting to log the coordinates on the maps.

Our proposed approach of learning IPS accuracies, referred to as the ***Learning Algorithm (LA)***, is currently implemented in Matlab. We compare our approach with the following three algorithms:

(a) ***Oracle Algorithm (OA)*** has access to the ground truth and can directly compute the actual accuracy $\epsilon_{oracle}(j)$ of an IPS at location l_j.

(b) ***Report-based Algorithm (RA)*** estimates accuracy based on reported observations z_t from a positioning system. The algorithm treats the location l_j with the largest probability according to z_t as the actual position of the user, and estimates IPS accuracy as the expected distance between the observation z_t and location l_j.

(c) ***Fusion-Based Algorithm (FA)*** takes observations from all co-located IPSs into account. For M IPSs, at time t the algorithm finds the centroid location l_j according to the observations $z_t^1, ..., z_t^M$. Each IPS then estimates its accuracy as the expected distance between its own observation z_t and location l_j.

We evaluate the above competing algorithms against the following two metrics:

(1) ***Estimation Error*** EE. For an accuracy estimation algorithm, EE is defined as the mean squared error between the estimated accuracy ϵ and the ground truth accuracy ϵ_{oracle}: $EE = \frac{1}{N}\sum_{j=1}^{N}(\epsilon(j) - \epsilon_{oracle}(j))^2$. We use EE_L, EE_R and EE_F to denote the estimation errors of *LA*, *RA* and *FA* respectively.

(2) ***Localization Error*** LE. Given estimates of IPS accuracy at different locations, we can partition the space into regions, where an IPS is consistently better than the others. The user can then choose the best available IPS in a given area. As different algorithms provide different estimates of IPS accuracy, they partition the space differently, and suggest a different IPS usage. They thus have a different localization error, which is defined as the mean squared error between the ground truth $\{x_i\}$ and estimated

trajectory $\{\hat{x}_i\}$: $LE = \frac{1}{T}\sum_{t=1}^{T}(\hat{x}_t - x_t)^2$. We use LE_L, LE_R and LE_F to denote the localization error when space partitioning is based on *LA*, *RA* and *FA* respectively.

4.2 Experiment Results

IPS Accuracy and Ranking Varies Over Space: The first experiment shows that the accuracy of a positioning system varies significantly over space, and that there may not exist a clear winner, i.e. an IPS that is consistently better than the others at all locations. In this experiment, three different IPSs ps_1, ps_2 and ps_3 are running in parallel on the 4th floor. As shown in Fig. 3(a), ps_1 has most of its nodes at the bottom left corner, ps_2 dominates on the right side, and ps_3 on the top side of the floorplan. Figs. 4(a1)∼Fig. 4(c1) show the estimated trajectories and reported accuracy (ellipses) when each of the three IPSs is tracking a user along the corridor. Figs. 4(a2)∼Fig. 4(c2) show the real accuracies of the three positioning systems at different locations. It is obvious that none of the IPS is globally better than the other, and each one tends to excel in the area where it has the denser infrastructure. Fig. 4(d1) shows the most accurate IPS at each trajectory location. To avoid switching too often, we transform Fig. 4(d1) to Fig. 4(d2), by applying a clustering algorithm that finds broader regions dominated by an IPS. Such a space partitioning would be useful to decide which IPS to use where.

Estimation Performance: The goal of this experiment is to evaluate how the estimation error of our algorithm (*LA*) fares compared to that of competing algorithms (*RA* and *FA*). In this experiment we use the same set up as in the first experiment, i.e. the three co-located IPSs on the fourth floor of a CS building. Fig. 5(a) shows the estimation error EE_L of our proposed algorithm at the end of each learning iteration. Whereas the error initially decreases as expected, it starts to increase again after the third iteration. This is due to the common problem of overfitting. We address this problem by randomly selecting a set of IPS observations (data collected in 5 days) as a training set that we use for learning, and a test set (data collected in another 5 days) that we used to detect when to terminate the learning process. As shown in Fig. 5(b), we terminate learning in the third iteration when the log likelihood of the test data starts to decrease. Fig. 5(a) shows that, when we pause learning in iteration 2, the error of the proposed algorithm (EE_L) is 3-4 times smaller than that the fusion-based error EE_F, and eight times smaller than the report-based error (EE_R). Fig. 5(c) shows that the estimation error of the proposed algorithm (*LA*) can be further improved significantly by incorporating prior information on the location of the user, which can be drawn from user calendars. The x-axis shows the percentage of timestamps for which we have prior information. As expected, the more the priors on user positions, the smaller the estimation error of our algorithm. For example, when priors are available at 10% of the timestamps, the estimation error of *LA* is $1.51m^2$, which is about 30% smaller than the case where no prior is used ($2.14m^2$).

Ranking Performance: In the previous experiment, we showed that the proposed algorithm outperforms competing approaches in terms of estimating IPS accuracy. We now show how this affects the user's ability to rank IPSs, and to choose the best one at each location. We examine four different scenarios depicted in the four rows of Fig. 6 respectively. The first scenario (row 1) includes positioning systems ps_1, ps_2 and ps_3

Fig. 3. (a) Experiment setup on the 4th floor. (b) Experiment setup on the basement.

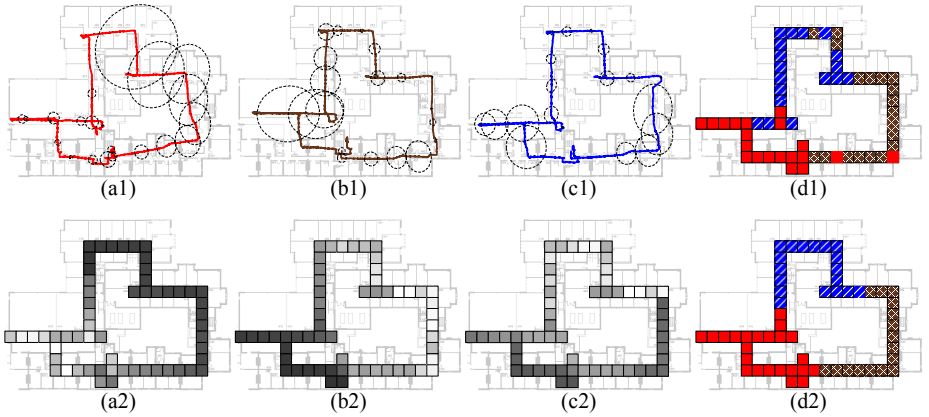


Fig. 4. (a1) Estimated trajectory from ps_1. (a2) Real accuracy of ps_1 (darker means less accurate) (b1) Estimated trajectory from ps_2. (b2) Real accuracy of ps_2 (darker means less accurate) (c1) Estimated trajectory from ps_3. (c2) Real accuracy of ps_3 (darker means less accurate) (d1) Each traversed location is labeled with the locally most accurate IPS. (d2) Space is clustered into regions where each of them is dominated by an IPS.

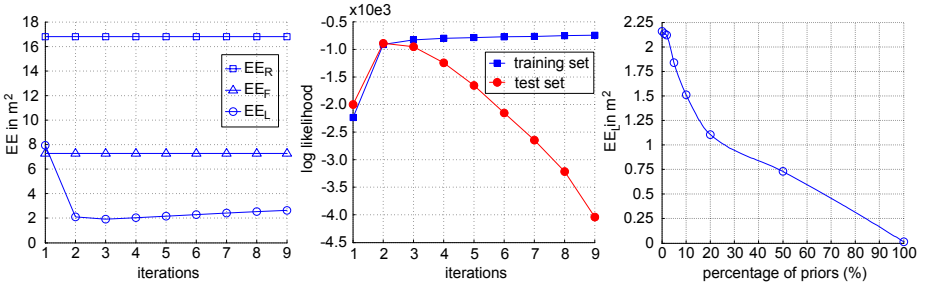


Fig. 5. (a) Log likelihood of data (both training and test) at different learning iterations. (b) Estimation error of *RA*, *FA* and *LA*. (c) Estimation error of the proposed *LA* as we vary the amount of priors used in the algorithm.

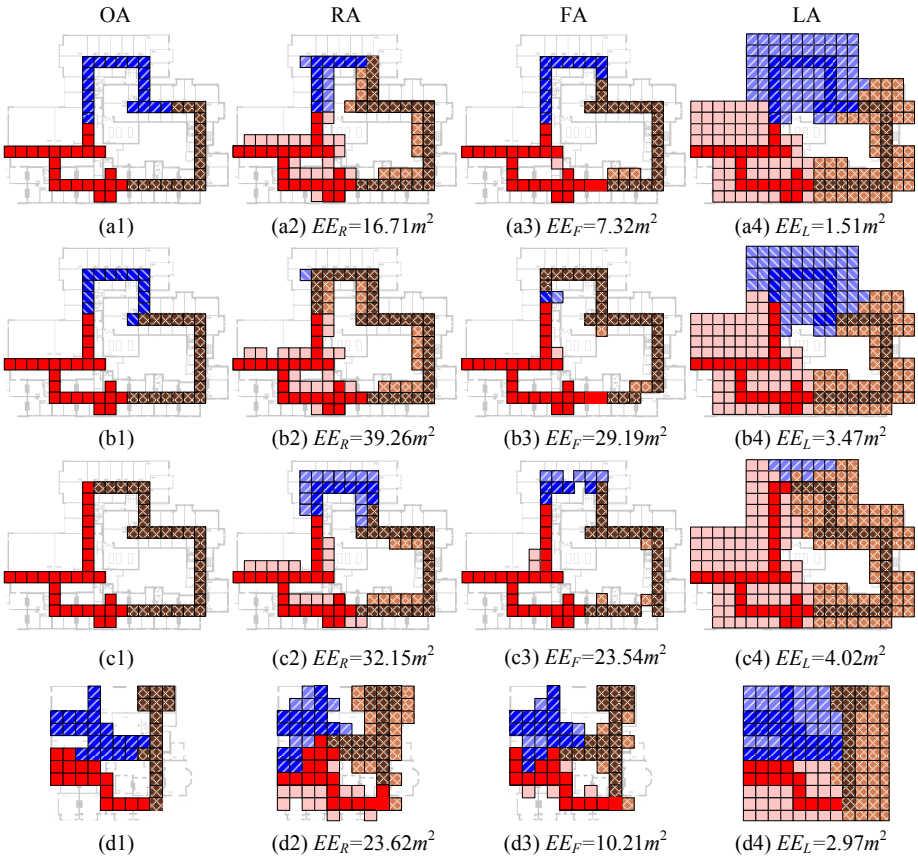


Fig. 6. Maps showing IPS preferences and estimation errors of different algorithms: *OA*, *RA*, *FA* and *LA*, with the trajectory locations highlighted and empty blocks indicate no information there. Row 1: Loc = 4F, User = Nexus S, IPSs = $\{ ps_1, ps_2, ps_3\}$; Row 2: Loc = 4F, User = Nexus S, IPSs = $\{ ps_1, ps_2, ps_4\}$; Row 3: Loc = 4F, User = TF201, IPSs = $\{ ps_1, ps_2, ps_3\}$; Row 4: Loc = 0F, User = Nexus S, IPSs = $\{ ps_5, ps_6, ps_7\}$. In all scenarios we assume for 10% of the timestamps we have prior information on locations of the user, which is used by *LA*.

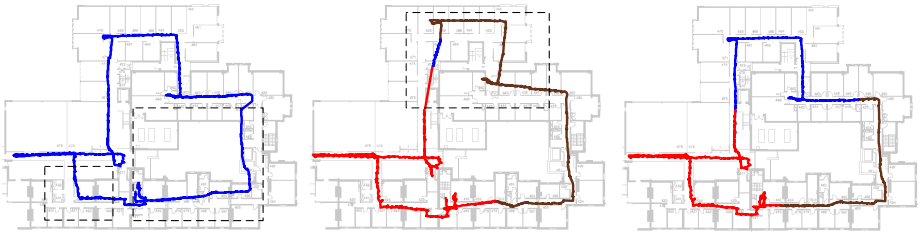


Fig. 7. (a) Trajectory from the single best IPS. Dashed rectangles indicate areas where localization error is larger than $4m^2$. (b) Switching according to *FA*, and $LE_F = 5.33m^2$. (c) Switching according to *LA*, and $LE_L = 2.07m^2$.

deployed on the 4th floor and a user traversing it holding a Nexus S device. Observe that the proposed algorithm (Fig. 6(a4)) manages to partition the space in almost the same way as the oracle (Fig. 6(a1)) that has perfect knowledge of IPS accuracies. In this scenario, the two competing algorithms (*RA* and *FA*) perform reasonably well (Figs. 6(a2) and 6(a3)), with the reported approach making a few errors in the upper part of the user's trajectory. In the second scenario (row 2), we use a different set of co-located positioning systems: ps_1, ps_2 and ps_4, where ps_4 has the same infrastructure as ps_3 but more conservative noise profile (with twice as high gyroscope and accelerometer variances). Although this does not change the ground truth much, it throws off the two competing algorithms (*RA* and *FA*), which now refrain from using ps_4 on the top part of the trajectory. Notice however that the proposed (*LA*) approach still perceives ps_4 to be the best there and still matches the *real* partitioning. In the third scenario, (row 3) we use the same setup as in the first scenario (row 1), except that the user now carries an Asus TF201 tablet instead of the Nexus S phone. Whereas ps_1 and ps_2 use different radio propagation models for the two devices, ps_3 has been tuned for phone users only. Hence, it performs poorly on the tablet and ceases to perform well on the top part of the user's trajectory (see Fig. 6(c1)). Notice that only the proposed algorithm (*LA*) manages to detect the drop in ps_3's accuracy and partition the space correctly, whereas the other two algorithms (*RA* and *FA*) still prefer ps_3 on the upper part of the trajectory. In the fourth scenario (row 4), we change the venue to the basement, which has more open areas than the the 4th floor. Here, the competing algorithms (*RA* and *FA*) erroneously show a preference on using ps_6 not only on the right hand side of the floor plan, but also further into the center (Figs. 6(d2) and 6(d3)). The proposed algorithm (*LA*) (Fig. 6(d4)) again partitions the space as if it had access to the real IPS accuracies (Fig. 6(d1)).

Localization Performance: The final experiment is set up to investigate how ranking performance has a knock-on effect on localization performance. That is, if a user relies on the space partitioning generated by the proposed algorithm, to switch from one IPS to another, will they be localized more accurately, than if they used that of the competing algorithms (*RA*/*FA*)? Here we compare only with *FA*, since it has proved to be superior to *RA*. We also compare with the strawman approach of using one IPS everywhere, the one that has the best overall performance in terms of localization error LE. Fig. 7 shows that switching IPSs is generally better than using the same IPS everywhere. It also shows that the localization error of the user that relies on the space partitioning of the proposed algorithm ($LE_L = 2.07m^2$) is reduced by more than 60% when compared to that of the best competing algorithm ($LE_F = 5.33m^2$).

5 Related Work

In this section we overview existing work and discuss how it relates to our problem. A large body of research in sensor networks has applied Bayesian State Estimation in localization- and tracking-related applications. For example, HMMs are commonly used to find the most likely trajectory that accounts for measurement noise and known map constraints (e.g. in VTrack, EasyTracker and CTrack systems) [6,10,15,11,1]. In addition, Bayesian filters, such as Kalman and particle filters have also been broadly used for online position estimation both in sensor networks and robotics research [3,14].

In stark contrast to the aforementioned HMM-based and filter-based approaches, we do not assume knowledge of the emission probabilities (a.k.a. observation models) of the systems used for localization. In addition, we have no ground truth data from measurement campaigns, which we could use to empirically learn the emission probabilities. Whereas they address the *inference* problem of estimating the most likely trajectory of a mobile node, we address the *learning* problem of estimating the IPS emission models that best explain their observations.

Another related area of research is that of cost-constrained data acquisition in sensor networks. Most work in this area explores the trade-off between the quality of information and the costs incurred in retrieving the sensor data, and it can be broadly divided into two branches. The first branch assumes that sensor readings concern the same variable of interest and are typically correlated over space. The goal is then to place sensors, or select sensors to be active, at informative locations, e.g. using Gaussian Process (GP) models of the sensed phenomenon [8,2,9,4]. The second branch of cost-bounded data acquisition considers heterogeneous sensors, and applications where the variable of interest may not be directly measured by a sensor, but rather derived from a variety of sensor measurements using a prediction model. An example is the study of selecting suitable regression models to minimize the prediction error given a cost budget [16]. Our work is different from the above cost-constrained data acquisition techniques in that we do not possess knowledge of the true locations where the noisy measurements were made, and thus cannot use it to train Gaussian Process or other regression models.

Our work is also related to fact finding techniques in information networks [7,19,12]. In these networks, sources and assertions are represented as nodes, and each fact "source i made an assertion j" is represented by a link. Nodes are then assigned credibility scores in an iterative manner: for example, in a basic fact finder [7], an assertion's score is set to be proportional to the number of its sources, weighted by the sources' scores; similarly, a source's score is set to be proportional to the number of the assertions it made, weighed by the assertions' scores. Whereas we share the same goal of assessing the credibility of different sources (in our case, IPSs), we cannot directly apply fact finding techniques for three reasons: First, the credibility of an IPS is not the same across different locations, which means that it does not make sense to try to evaluate a single credibility score for an IPS across the entire area of interest. Second, the true user location is not known, so we cannot define a source, as an (IPS,location) pair. Finally, fact finding techniques tend to work well when a large number of sources are used (e.g. social sensing), whereas in our case our goal is to assess and rank a limited number of positioning systems (typically, less than 5).

Our work falls into the class of maximum likelihood estimation (MLE) approaches to ascertaining sensor reliability. In this sense, it bears close resemblance to [17], which estimates both the correctness of measurements and the reliability of participants in social sensing applications by solving an expectation maximization problem. Similar to [17], we also adopt an expectation maximization (EM) approach to get the maximum likelihood estimate of model parameters. However, the underlying probabilistic model, the problem formulation, and the specific derivation of the EM algorithm are very different. Unlike [17], we consider a dynamic model, with probabilistic observations, and

cast our problem in that of learning the parameters of a HMM. We thus use a different EM scheme (an extension of Baum-Welch algorithm), which is suitable for HMMs.

6 Conclusion and Future Work

This paper contributes a novel approach of estimating the accuracies of co-located IPSs. The estimation problem is cast into parameter learning of an augmented HMM, and solved by a variation of the EM algorithm. We show that prior information provided by the users can significantly improve the quality of estimation. Experiments in four indoor scenarios shows that our approach outperforms the competing ones both in terms of estimation and localization error. Future work includes supporting continuous observations and on-line learning, utilizing more types of priors, and working with more complex positioning systems, whose performance does not only vary with space.

Acknowledgements. The authors would like to acknowledge the support of the EPSRC through project FRESNEL EP/G069557/1, and thank the anonymous reviewers and our shepherd Philippe Bonet for their helpful feedback.

References

1. Biagioni, J., Gerlich, T., Merrifield, T., Eriksson, J.: Easytracker: automatic transit tracking, mapping, and arrival time prediction using smartphones. In: Proc. SenSys (2011)
2. Das, A., Kempe, D.: Sensor selection for minimising worst-case prediction error. In: Proc. IPSN (2008)
3. Fox, D., Hightower, J., Liao, L., Schulz, D., Borriello, G.: Bayesian filtering for location estimation. In: IEEE Pervasive Computing (2003)
4. Hare, G.M.P.O., Muldoon, C., Trigoni, N.: Combining sensor selection with routing and scheduling in wireless sensor networks. In: Proc. DMSN in Conjunction with VLDB (2011)
5. Huang, J., Millman, D., Quigley, M., Stavens, D., Thrun, S., Aggarwal, A.: Efficient, generalized indoor wifi graphslam. In: Proc. ICRA (2011)
6. Hummel, B.: Map matching for vehicle guidance. In: Dynamic and Mobile GIS: Investigating Space and Time. CRC Press (2006)
7. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. Journal of the ACM 46(5) (1999)
8. Krause, A., Guestrin, C., Gupta, A., Kleinberg, J.: Near optimal sensor placements: maximising information while minimising communication cost. In: Proc. IPSN (2006)
9. Krause, A., Singh, A., Guestrin, C.: Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. J. Mach. Learn. Res. 9 (2008)
10. Krumm, J., Letchner, J., Horvitz, E.: Map matching with travel time constraints. In: SAE World Congress (2007)
11. Newson, P., Krumm, J.: Hidden markov map matching through noise and sparseness. In: Proc. GIS (2009)
12. Pasternack, J., Roth, D.: Knowing what to believe (when you already know something). In: Proc. COLING (2010)
13. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Pearson Education (2003)

14. Schmid, J., Beutler, F., Noack, B., Hanebeck, U., Müller-Glaser, K.D.: An experimental evaluation of position estimation methods for person localization in wireless sensor networks. In: Proc. EWSN (2011)
15. Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., Eriksson, J.: Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In: Proc. SenSys (2009)
16. Wang, D., Ahmadi, H., Abdelzaher, T., Chenji, H., Stoleru, R., Aggarwal, C.: Optimizing quality-of-information in cost-sensitive sensor data fusion. In: Proc. DCOSS (2011)
17. Wang, D., Kaplan, L., Le, H., Abdelzaher, T.: On truth discovery in social sensing: a maximum likelihood estimation approach. In: Proc. IPSN (2012)
18. Welch, L.R.: Hidden Markov Models and the Baum-Welch Algorithm. IEEE Information Theory Society Newsletter 53(4) (2003)
19. Yin, X., Han, J., Yu, P.: Truth discovery with multiple conflicting information providers on the web. In: Proc. KDD (2007)

TrainSense: A Novel Infrastructure to Support Mobility in Wireless Sensor Networks

Hugues Smeets, Chia-Yen Shih, Marco Zuniga, Tobias Hagemeier, and Pedro José Marrón

Universität Duisburg-Essen, Germany
{hugues.smeets,chia-yen.shih,marco.zuniga,
tobias.hagemeier,pjmarron}@uni-due.de

Abstract. In this paper, we present *TrainSense*, a novel infrastructure to support the development and testing of mobile sensing applications. TrainSense merges a mote and a model train into a single mobile unit, and enhances the basic model train infrastructure with several important features required for the evaluation of mobile scenarios. First, we develop a *real-time* controller to send control packets to model trains and motes to manage the network topology. Second, we design and implement a positioning system with centimeter precision. Third, we use the power available on the tracks to provide unlimited energy to the motes. Fourth, we provide a way for the motes to dock into a custom USB port, for reprogramming and data download. We evaluate TrainSense in two ways: (i) we establish the correctness of the implementation and measure the performance of its components, and (ii) we demonstrate its practical use with two sample wireless sensor network application scenarios: self-deployment and data muling.

1 Introduction

Several studies highlight the fundamental role that mobility plays on the performance of Wireless Sensor Networks (WSNs). Mobile nodes can, for example, increase the communication capacity [12], enhance the sensing coverage [19] and facilitate network deployment [26]. But in order to achieve these benefits, the underlying mobile infrastructure has to provide some important features. Mobile WSNs need (i) advanced *topology management* capabilities, i.e. the ability to specify, simultaneously, the speed and direction of each individual node; (ii) the *ability to track and localize* nodes; (iii) a *reliable source of energy* to avoid unnecessary pauses or abrupt stops; and (iv) *automatic operations* for re-programming and data download, to minimize the amount of manual overhead.

Robotic ground vehicles can provide several of the features mentioned above, and hence, they have been the primary choice to enable WSN mobile applications ranging from mapping and navigation [17] to search and rescue [16]. However, before a robot-based infrastructure is "WSN-ready", some technical challenges need to be tackled such as, path scheduling [25, 13], localization [8], energy budgets, and deciding on suitable hardware platforms to sustain the particular

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 18–33, 2013.

demands of the application [23, 3]. Depending on the researcher's expertise, overcoming these challenges may not be an easy task.

Our aim is to propose an alternative platform to evaluate mobile WSNs applications. We believe that model trains have the necessary capabilities to support the development and evaluation of such applications. The state of the art in model trains provides sophisticated hardware for topology management. Train controllers can address multiple trains, through the tracks, to modify their individual speed and direction. This basic infrastructure has important characteristics that can be leveraged. The tracks can be used (i) to provide infinite energy to the mobile motes, (ii) to detect the position of nodes and (iii) as a back-up channel to send information to the motes. These features can help in testing various mobile scenarios. For example, a pre-deployment evaluation requiring a systematic study of the node positions could use TrainSense to move nodes sequentially to various locations and identify the deployment or mobility pattern that maximizes the metric of interest. The application could also use the tracks to simulate node failures by sending on/off commands to the motes, without interfering with the normal operation of the experiments.

In this paper, we propose TrainSense. The main idea of TrainSense is simple, but powerful: to integrate a model train and a mote into a single unit and to utilize the integrated mobile node along with the model train infrastructure for WSN applications. The main contribution of our work is the enhancement of the typical model train platform with the following new features to offer mobility support: *real-time train control*, *precise train positioning*, *energy management*, *automatic train-mote operations* and *back-channel information support*.

2 TrainSense Infrastructure

In this section, we describe the basic model train infrastructure, and then, we provide a general overview of TrainSense.

2.1 The Basic Model Train System

The technology to control model trains digitally is available since 1986 and it was introduced by Maerklin as the Maerklin Digital System. Figure 1 outlines a basic off-the-shelf train system. This basic system consists of five main components: (i) **the host computer**, which provides a user friendly interface to define the train movements, (ii) **the controller**, which sends packets through the tracks to control the speed and direction of locomotives, and to control the turnouts, (iii) **the detector**, which identifies the presence of trains when they reach a particular track section and reports these events to the controller, (iv) **the tracks**, which carry power and data to the trains, and (v) **the locomotives**, each one having a decoder to process the commands from the controller.

The user has significant freedom to design a track layout. The various track elements can be combined into practically any shape with the only constraint being a minimum curve radius of 36 cm. To make the system operational, some

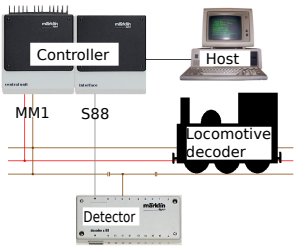


Fig. 1. The basic off-the-shelf train control system

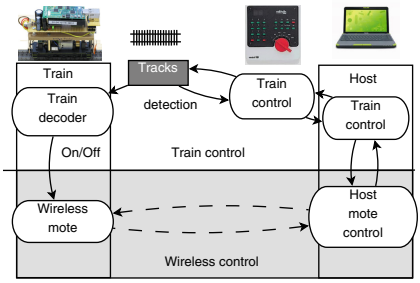


Fig. 2. The TrainSense architecture

simple wiring is required to connect the controller, tracks and detectors. The controller can be configured to handle simple automatic movements such as a train traveling between two points. If more complex mobility patterns are required, a computer with the appropriate software can be connected to the controller through an RS232 interface. A typical commercial system can control multiple trains and select their routes. It can also localize and position trains with a coarse precision, about 10cm. While these systems are well designed to reproduce the behavior of life-size trains, they miss important functions for the evaluation of WSNs. In the next section, we provide a broad description of the enhancements that we made to the basic system for supporting mobile WSNs.

2.2 TrainSense Architecture

Our TrainSense architecture introduces changes to the basic system at two levels: train control and wireless control levels (see Figure 2).

At the train control level, TrainSense includes the basic system with the typical wiring required for standard components. There are, however, two important differences. First, the hardware and software of the controller and the detection mechanism have been re-designed to meet the requirements of mobility-enabled wireless sensor networks. Second, we developed new software for the host computer to allow more elaborated mobility patterns, as detailed in Section 3.1.

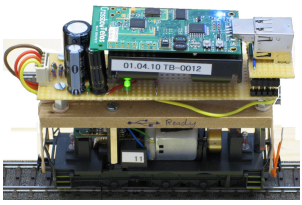


Fig. 3. A TrainSense mote

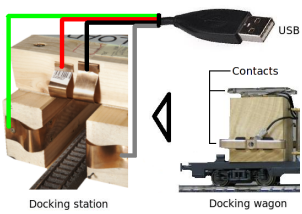


Fig. 4. The docking station

At the wireless control level, we introduced three new components for WSN applications: (i) the TrainSense mote, (ii) the host mote and (iii) the docking station. As illustrated in Figure 3, the TrainSense mote is composed of four main parts: a locomotive with its decoder, a mote, a custom decoder for the mote to receive data from the tracks, and the on-board power supply. To have a closed control loop over the train mobility, **the host-mote** connected to the host computer allows the mobile motes to send packets to change the speed and direction of other trains. Our infrastructure is operating system agnostic and can be used seamlessly with TinyOS or Contiki, for example.

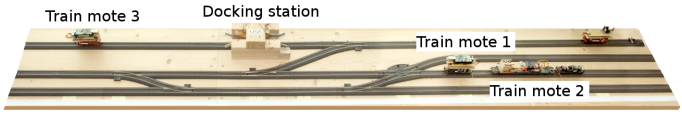


Fig. 5. The main segment of TrainSense

Finally, the **USB docking station** (see Figure 4) provides the same functionality as the wired USB connection, and has an effective bandwidth of about 97% of its wired counterpart. While connected to the docking station, a TrainSense mote can be reprogrammed or data can be downloaded automatically. Figure 5 illustrates the physical construction of a segment in TrainSense with the docking station and some motes.

3 TrainSense Features

TrainSense has several important features to support mobile WSNs. We now describe in detail their implementation and evaluation.

3.1 Real-Time Train Control

Real-time train control is the key feature to enable flexible, interactive topology management. Together with the host program, the TrainSense controller offers the possibility to communicate with motes through the tracks. It provides real-time guarantees for fine-grained topology management and accurate positioning.

Train Control Host Program. In WSNs, mobility patterns could be rather complex and a node may be required to make precise stops, or to change direction rapidly. For this purpose, we developed a new host software with various mobility primitives to support more complex mobile scenarios. The host program allows the user to specify commands for the train movement control and to define a handler action in response to detection events.

The Real-time TrainSense Controller. Several standards are available for model train communication. Our controller is based on the Maerklin/Motorola

standard [1] because (i) it is well documented, (ii) the latest formats are backward compatible, (iii) the hardware for this format is readily available, and (iv) it is simple to implement. The controller can address 80 mobile trains and control 256 turnouts. The standard defines two main types of packets that can be sent through the tracks: packets containing addresses, speed information and direction commands (for trains), and packets for the turnouts to switch left or right. The train packets take less than 20 ms to reach a train. The data rate of the turnouts packets is twice as high, requiring only 10ms to reach a turnout.

The commercially available train controllers are not real-time systems, which is problematic since we need a short upper bounded time to detect events in order to position trains precisely. Therefore, we designed our TrainSense controller to provide real-time detection and train control. The control services that can be guaranteed to be performed in real-time include: setting locomotive speed, turnout direction, trigger upon a detection event, and getting the time between the last triggered action and the monitored event. To facilitate network management, our controller can send new types of packets to configure the motes.

3.2 Precise Positioning

TrainSense provides precise positioning of motes at user-specified locations. Conceptually, the basic detection mechanism works as follows: when a train crosses a point that has a detector, the wheels of the train create a short circuit between the two rails. This event is registered in a hardware register. The controller continuously polls the detector-registers and identifies if a train reached a detection point. In regular train model applications this method is used to stop the train or to switch the direction of a coming turnout.

We extended this detection method with a dead-reckoning[1] technique to position nodes anywhere on the track: if a node crosses detector x at time t with speed s, the node can be positioned at $x + \Delta x$, by waiting $\frac{\Delta x}{s}$ seconds and then sending a "stop" packet to the train. However, for dead reckoning to be accurate, the controller must provide short and bounded delays for register reading and for data packet delivery.

To provide the required real-time guarantees, the TrainSense controller assigns the highest priority to data packet transmissions, and the second priority to register reading. Reading each register takes 200 μs, hence if there are n detectors, the controller requires $n \times 200\mu s$ to read all the registers. Usually, data transmission operations take 10% of the CPU time. Considering a train moving at the maximum speed of 0.5 m/s, the maximum positioning error introduced by the controller is $[(0.2n)1.1 + 20]ms \times 0.5m/s$. Assuming 50 detectors, the maximum error is ≈ 1.5 cm.

[1] In the rest of this paper, "dead reckoning" refers to the process of estimating the position of a mote by having it moving at a constant speed when it crosses the initial position and letting it run for a determined time to position it at a determined distance from that initial position. This requires starting the mote ahead enough of the initial position.

One issue with dead reckoning is the potential of cumulative errors, that is, the longer a mote travels before re-calibrating its location to a ground truth, the higher the positioning and localization error can be. The detectors and the tracks play the role of re-calibrating the dead reckoning estimation. Depending on the availability of a detector near the train, there are two types of positioning scenarios in TrainSense: detector-based or dead-start[2].

Detector-based Scenario. In dead reckoning, it is assumed that the train speed is constant, which is not always true. When a train starts from a still position, it takes approximately 1 to 2 seconds for the train to reach its nominal speed. This initial period introduces an error in dead reckoning. We eliminate this error if we can make the train reach its nominal speed before it reaches the detector used for the dead reackoning. To evaluate the positioning precision of that scenario, we performed the following set of experiments: a train was positioned 20 cm before a detector, which allowed sufficient time for the train to reach its nominal speed. After the controller detected that the train had crossed the detector, a stop packet was sent to position the train at $i \times s$ cm, where s represents the nominal speed of the train in cm/s and i represents a period of time that goes from 1 sec to 8 sec in steps of 1 sec. The tested nominal speeds are: 5, 7, 10, 15 which were translated to 12, 19, 30 and 37 cm/s. At each nominal speed, each point was measured 30 times.

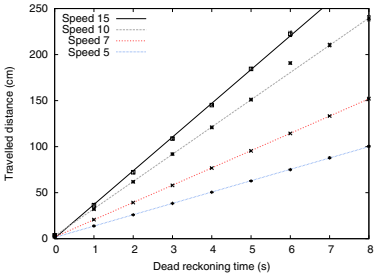


Fig. 6. Positioning the mote using the detectors and dead reckoning.

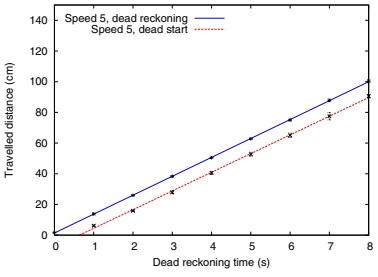


Fig. 7. Positioning the mote using dead start versus dead reckoning

Figure 6 shows the accuracy of the positioning method. Each point represents the average of the reached positions. Each line shows a linear regression on all the points corresponding to a given speed setting. There are two important points to highlight: first, for most points, the margin of error is below 2cm, and the maximum deviation, a bit more than 2cm, was obtained at the maximum distance traveled, due to cumulative errors of dead reckoning. Second, occasionally, the speed of the train changes and causes larger errors than expected (see

[2] When dead reckoning is not possible, the mote starts from zero speed, reaches constant speed after a certain time and is left moving at that speed for a determined time. We call this technique "dead start".

point 6 for speed 10). We found that this happens due to manufacturing errors, in some occasions the motor stutters and changes its nominal speed. This is not a common event, but it occurs and its effect on the accuracy of dead reckoning is within 10cm (for the distances and speeds measured). The solution is to have several detectors in the infrastructure to correct inaccuracies.

Dead-start Scenario. If there is no detector before the target point, or if the detectors are not far enough to allow the mote to reach its nominal speed, then TrainSense switches to a dead start scenario. The likelihood of dead-start events depends on the density of detectors in the infrastructure, the more detectors, the less frequent the event. Figure 7 shows a comparison between two scenarios for speed 5. We found that trains had more difficulty in overcoming their inertia at lower speeds. The evaluation process was the same as the one for dead reckoning but without using a detector. The mote was placed at position 0 and control packets were used to start and stop the train. We observe that dead start had a **constant** negative offset of about 10 cm, with respect to dead reckoning. This effect can be canceled by letting the train run for a longer period when no detector is found during the positioning trip (a 1 sec delay for this speed). The precision is therefore also 2cm, but the resolution is limited to 10cm. Dead reckoning is needed for the best resolution.

3.3 Energy Management

Energy consumption has been the main limitation of mobile networks. Model trains allow us to leverage the tracks to provide a constant energy source to the motes. However, several issues need to be solved. First, to extract energy from the tracks, we built a classic AC to DC converter to convert the voltage to 5V from an alternative voltage of ± 18V. The converter resides on the **on-board power supply** unit. It takes the track voltage as input, and regulates and buffers the output to power the mote through the mote's USB interface. The schematic of the whole on-board electronics (power supply, mote control and mote) is shown in figure 8.

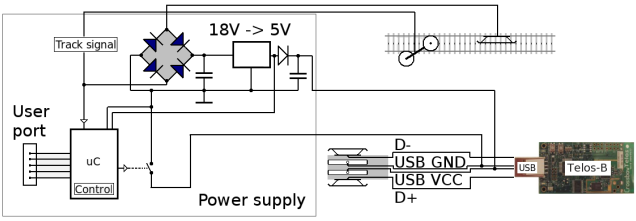


Fig. 8. The schematic of the power supply

The second issue is the lack of constant connectivity, i.e., if the tracks become dirty, the power supply is temporarily cut off, reseting the mote. To guarantee

a constant power supply, we use gold-caps, which are small capacitors with very high capacity to make the power supply resilient to contact problems. We must ensure that the power supply is not affected by the short disruption period caused by dirty tracks. To assess the reliability of the power supply, we measured the duration of operation without power from the tracks during which we can still provide a stable voltage to the connected mote. Given that communication is the most energy-consuming activity in a mote [28], the evaluation is done with the radio continuously on. We programmed a mote to send packets constantly at the minimum and maximum power levels: -20dBm and 0dBm, respectively. The mote was placed at approximately 2 meters from the host-mote. The host-mote measured the RSSIs of the received packets. During the experiments, we measured the output voltage of our on-board power supply with an oscilloscope. Once the mote was positioned on the tracks, we charged the gold capacitors for a minute using the baseline mode (i.e., a constant 18V in the track with no activity on the tracks). Then, we disconnected the power supply from the tracks. The results are shown in figure 9. As we can see, even at the maximum transmission power level, the capacitors provided a lifetime of 1.5 minutes after disconnection (the point at which the RSSI drops drastically). For the lowest power level, the lifetime was almost 3 minutes. Usually, power disconnections caused by dirty tracks occur for at most fractions of a second, and hence, the on-board power supply guarantees continuous power to the mote.

Another useful feature is the **on-board mote control**, which allows turning the motes on and off remotely. This feature can simulate a full or an empty battery. An 89C51 Atmel controller decodes the address field in the track signal. If it matches its own, it reads the function bit: a one/zero turning the mote on/off, respectively. Our controller allows making use of the data fields to include other commands e.g., measuring the mote's power consumption.

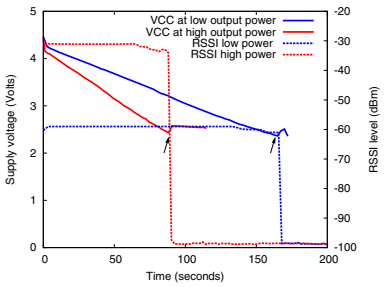


Fig. 9. The power supply voltage

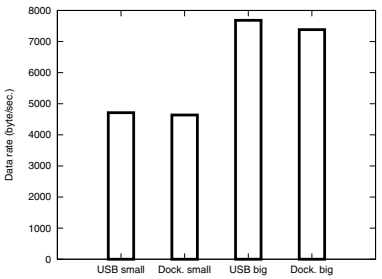


Fig. 10. Docking vs. Direct USB

3.4 Automatic Reprogramming and Data Download

To facilitate re-programming and data downloading, we developed USB docking stations for the TrainSense mote as described in Figure 4. The physical docking station is simple and inexpensive. It has four contacts, which are made of

"chrysoscale", an alloy whose electric conductivity is as good as copper, but it does not rust. The two contacts on the top are used for the power (USB VCC and ground) and the two on the side are the data lines. The goal behind the design of the USB docking station was to provide the reliability, bandwidth and zero-interference effect of a wired USB port. However, since the trains do not have the power to dock a mote directly into a regular female USB port, the docking station was developed to allow easier, automatic insertion.

To evaluate the throughput of our docking station, we positioned a TrainSense mote 10cm from the station and used the controller to dock in the mote. The mote was then instructed to send 1000 packets of 10 bytes and 1000 packets of 50 bytes. After that, the mote was moved to the original un-docked position, and the docking was repeated 10 times. We compared the transmission taking place through our docking station with transmissions on a standard 1.5m long USB cable. Figure 10 shows our test results.

For the docking station, the data rate of the small packets was 98% compared to the regular port, for the 50-byte packets, 96%. It is important to note that no packets were lost: the failed transmissions were successfully corrected at the USB level, which has a robust error correction mechanism with retransmission. The lost packets simply result in a reduced transmission rate. This indicates that the connections of our docking station are not exactly as good as a standard cable, but they have a negligible impact on performance.

3.5 Back-Channel Information

As mentioned before, an important feature of model trains is that the tracks are used to convey control data. Therefore, each train has a decoder that can be extended to communicate with the motes. To send a command to a node, the controller sends a packet to the address of the locomotive that carries the mote. Once the mote-decoder identifies its address, it performs the required action. Currently, we have implemented a mote on/off command to control the topology dynamically.

The back channel design is a good example of the modularity of the TrainSense architecture. Since we use the standard packet format, a commercial controller could also send information to the nodes. For example, if an application does not require the real-time characteristics of TrainSense (i.e., no fine grained positioning), a user could just purchase a commercial system and connect the on-board card (Figure 3) to the train and mote.

3.6 Zero Electromagnetic Interferences

As described previously, the tracks are used to provide power to the trains and to activate the turnouts. When the trains are moving or turnouts are activated, they draw currents whose flow in the rails causes an "antenna effect", i.e., generating electromagnetic fields that can cause interference to radio frequency communication. It is important to asses the impact of such an "antenna effect". We show

in this section that the electromagnetic fields generated by the tracks do not cause any degradation in the communication among motes.

We evaluate four operational modes. **Mode A–baseline**: the tracks are powered using the on-board power supply with 18 volts DC and no interference signal is present. This condition resembles a mote operating with ideal batteries (3V) without TrainSense. **Mode B–heavy traffic**: the controller sends a continuous stream of packets to the trains. This condition resembles a heavy traffic operation of TrainSense. **Mode C–frequent turnouts changes**: the turnouts are switched back and forth as fast as possible. Frequently switching turnouts is the harshest condition of all, because the coils within the turnouts consume more than one ampere. Hence, they can produce high electromagnetic pulses. **Mode D–dirty tracks**: the power on the tracks is turned on and off every second. This condition is very unlikely to happen during normal operation. It resembles extremely dirty tracks causing repeated loss of electric contact.

In order to evaluate the interference level, we performed two sets of experiments. In both experiments, we made sure that no other source of interference is present, e.g., WiFi. First, we positioned a TrainSense mote nearby four turnouts and measured the noise floor while the controller activated the four different modes. The noise floor was sampled every 500 μs (at 2KHz) and the four modes were tested in a round robin fashion with 10 seconds for each mode. A total of 1000 rounds were measured. Figure 11 shows the distribution of the measured noise floor values for each mode. Each distribution contains 20 million noise floor samples. We can observe no significant differences.

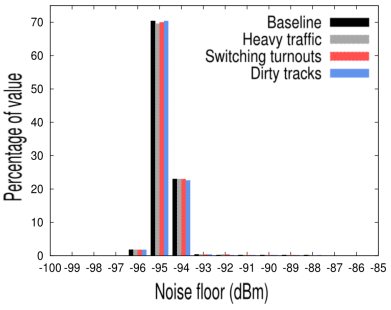


Fig. 11. The noise floor measurement

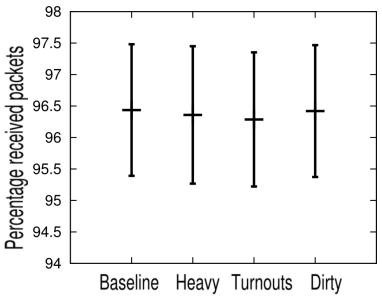


Fig. 12. Packet loss measurement

The noise floor measurement is a good indicator that the sampled noise floor remains constant during the different modes, but this is not sufficient. The motors of the trains, the coils of the turnouts and the sudden changes on the DC voltage of the tracks may cause pulses of short duration (in the order of a few microseconds) that are missed by the relative coarse sampling rate of 2KHz. The second set of experiments measures the effect of these short pulses.

Increasing the sampling rate is not the best option to capture short interference pulses. Even at the highest noise sampling rate achieved for motes, the short

pulses would be missed (60KHz in [5]). Thus, we used actual packet transmissions to quantify the effect of such interferences. The hypothesis is the following: short interference pulses can corrupt a few bits of an ongoing packet transmission. Hence, when they are present we should see a decrease of the packet reception rate. To test this, we used the same setup as in the previous experiment, but made the mote send packets continuously. We used a controlled scenario to make sure that the packet reception rate was not affected by dynamics in the environment. We used the lowest output power (-20dBm) and selected a distance that lead to a low RSSI at the receiver, so any variation in the noise floor could be detected via a packet loss. Short pulses would damage some bits of a packet, and hence, cause a packet loss. Figure 12 shows the mean and standard deviation of the packet delivery rate for the four operation modes. We observe that there is no statistical difference, the difference between the best and worst modes being negligible (less than 0.2% of the reception rate).

4 Supporting Mobile WSN Applications

In this section we demonstrate the support offered by TrainSense to implement two sample mobile WSN applications: self-deployment and data mules. Both applications use the positioning system and the constant energy features. In the data mule case, we also use the USB docking station. Both scenarios are implemented using TelosB motes and TinyOS 2.1.

4.1 Self-deployment Application

In self-deployment applications, nodes move autonomously in the target area to identify the locations that maximize a network metric, e.g., coverage or connectivity. Our application aims to evaluate a route-repair scenario (connectivity) using self-deploying motes to repair the route between two disconnected nodes.

Set-up. We place two nodes, a source and a sink, statically at the opposite extremes of the TrainSense prototype as shown in Figure 5. The sink node is connected to the host computer as the host-mote. Recall that the host-mote can interact with the controller to control the train movement. Given the short distance between the sink and source (around 3m), we use a pseudo-signal attenuation to achieve multi-hop transmissions. A link is considered valid if its RSSI is above -75dBm. For an output power of 0dBm, this threshold leads to a transmission range between a few tens of cm up to 1m. On the source side, we place six mote-trains. The mission of these motes is to establish a multi-hop path between the source and the sink.

We use a greedy approach for data forwarding. The data is forwarded to the mote that is closest to the sink. We use the mote ids to represent sink-proximity: the lower the id, the closer the mote is to the sink. The six mote-trains are aligned with monotonically decreasing ids (towards the sink). Before sending a packet, the source node sends a neighbor discovery message. Each neighbor replies with its id and the RSSI of its incoming link. To avoid collisions, the motes

spread their transmissions using a simple TDMA-MAC based on their node-ids. Upon receiving the reply packets from its neighbors, the source forwards the information to the neighbor that is closest to the sink and that has a link above the specified threshold. All intermediate motes function the same way.

Deployment. Initially, the sink detects that no packets arrive from the source and, thus, instructs the train controller to deploy the first mote-train with the lowest id. The first train moves towards the sink sending continuous discovery beacons. When the RSSI between the mobile mote and the sink reaches a value above the required threshold, the link is established. Then the sink instructs the controller to stop the train. If the mote does not repair the route, i.e., the sink still receives no packets from the source, the sink instructs the controller to deploy the next train, and this process is repeated until the route is repaired.

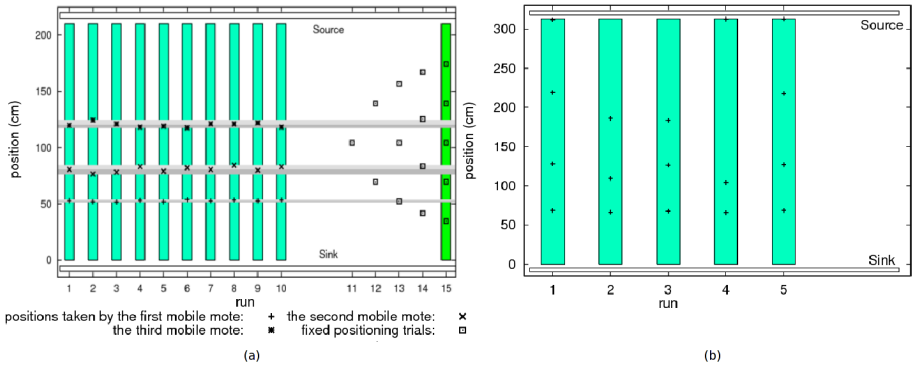


Fig. 13. (a) A comparison of automatic and fixed deployment, (b) Self-deployment in an uncontrolled environment

Results. We perform our evaluations in two types of environments: controlled and uncontrolled. In the controlled scenario, to obtain consistent results, we remove potential sources of interference: leaving the lab unoccupied and closing the door to avoid dynamics in the wireless channel. The experiments are performed remotely through the host computer. We perform ten runs of the experiment, and the results are presented in the first ten points of the x-axis in Figure 13(a). The figure shows that three motes are required to create a connected chain. The shaded areas indicate the minimum, maximum and average positions of these motes. The low variance in the final positions reflects the low channel dynamics. For completeness, we also perform uniform deployments (shown in the last five x-axis points of the same figure). The aim is to test the other end of the self-deployment spectrum, where nodes are deployed in a uniform way (same inter-node distance). In this case, the controller instructs n nodes to deploy in a uniform manner. The uniform deployment is achieved by using the positioning feature of TrainSense. The value of n varies from 1 to 5, in steps of 1. For each n, we perform ten trials. The path is repaired with at least five motes.

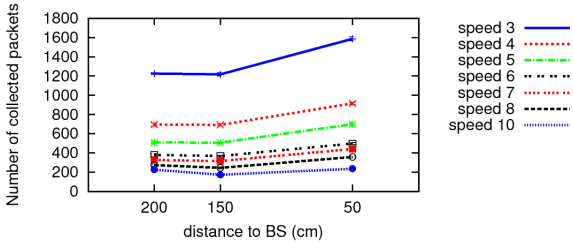


Fig. 14. Comparing data collection at various constant speeds

In order to assess the dynamics of the environment, we also perform the self deployment experiment in an uncontrolled scenario, where people move across the room. The same experimental setting is used except for the distance between the source and the sink (3.10 meters instead of 2.2). Figure 13(b) shows the significant effect of the room conditions on the number of nodes required to repair the path and their locations. This behavior is somehow expected due to the well-known fading effects in wireless communication, where shifts of a few centimeters (half the wavelength) can change the signal strength by tens of dBm.

4.2 Data Muling

We present now a data-mule application that makes use of the docking station offered by TrainSense. The aim is to have the data mule to move along the track, collect data from some static nodes placed on the side of the tracks and then to offload the collected data at the USB docking station.

Setup. Three static motes are placed at 50, 150 and 200cm from the docking station located at one end of the TrainSense track. These static nodes are positioned 20cm away from the tracks and they broadcast their data with power level 1 (chip CC2420, TelosB motes). The data mule process is started by the host using the train controller. The data mule makes one round trip, toward the opposite end of the infrastructure and back to the docking station. Power level 1 is not very reliable, but it is appropriate for this scenario because we do not test repeatability and the transmission range is about 30cm.

Results. Figure 14 depicts the results of the data mule experiment with various speed levels. The x axis indicates the location of the static nodes and the y axis the number of packets collected. All collected packets were correctly delivered through the USB docking station. The figure shows the expected trade-off between speed and data collection, the lower the speed the higher the collected packets. For example, for speed 5 (12cm/s), the number of collected packets per node is approximately 500, while for speed 10 (30cm/s), the number is approximately 200. When normalized with respect to the round trip time, both speeds lead to the same delivery performance. In practical settings however lower speeds are preferred because they would reduce the number of docking operations, which allows more time for data-gathering.

5 Related Work

In this section, we discuss related work with respect to mobility support, existing testbeds and mobility-assisted network deployment for WSN applications.

Exploiting node mobility is a good way to improve important network performance such as lifetime and throughput [27, 21, 4]. Node mobility can be classified into *uncontrolled* and *controlled mobility* [4]. With uncontrolled mobility, a mobile node can move randomly and freely. In contrast, with controlled mobility, a mobile node only moves along the pre-defined trajectory. Several studies have shown that controlling the mobility of the node (e.g., a mobile sink) can achieve remarkable performance improvement, especially on the WSN lifetime. The concept of controlled mobility was first presented by Gandham et al. [11]. The author presented an ILP (integer linear program) approach to deploy multiple mobile stations to prolong the WSN lifetime. Much work has been conducted on WSNs with mobile nodes moving along fixed paths [15, 18, 21, 4]. In [20], the author proposes a perimeter-based solution, in which the mobile sink cyclically moves along the perimeter of the monitored area. In [24], Somasundara et al. proposed a mobile infrastructure to reduce the communication energy consumption of low-energy embedded devices. TrainSense offers a model-train-based infrastructure that can support these controlled mobility scenarios.

Since TrainSense can also serve as a testing environment, we include related work on the mobility testbeds. MiNT-m [6] is a well-known testbed that provides physical mobility support. The mobile platform is based on Roomba [2] robots fitted with unique multi-color tags that allow localization. With respect to MiNT-m, the advantages of TrainSense are constant energy supply, cost (a train is approximately 3 times less expensive than the most affordable robots). The main comparative disadvantage of TrainSense is the restricted movement and the initial setup. TrueMobile [14] is a testbed where a few mica2 nodes are carried by robots. The robots are tracked from above using a vision system. On a similar line of work, Foerster et al. [9] present an architecture where nodes are carried by robots moving on the ground. A camera keeps track of the robot positions. These robotics testbeds share advantages and disadvantages that are similar to MiNT-m, when compared to TrainSense.

TrainSense is a unique mobility infrastructure. To the best of our knowledge, there is no prior effort that takes a model train setup and enhances it to the extent presented in this study. There are, however, studies that exploit the basic functionality of integrating a mote with a train. Roziers et. al. provide a testbed where a single node is located on a train that circles around a 3m×3m area [7]. Similarly, Rensfelt et. al. developed a simple robot that can follow a line painted on the floor [22]. In SmartHop [10], the authors use model trains to evaluate hand-off mechanisms in sensor networks. TrainSense also focuses on constrained paths, but provides significantly more advantages such as continuous power, complex topology management and positioning, and a USB docking station.

6 Conclusion and Future Work

The main goal of our study is to bring to the attention of the community a new platform to test and debug mobile WSNs. The new platform, TrainSense, is based on off-the-shelf model trains. The inherent characteristics of model trains – controllers, powered tracks and detectors – provide a natural configuration to support mobile WSNs applications. We took this basic infrastructure and added important features to create a comprehensive mobility platform. TrainSense provides continuous and unlimited energy to the motes, allows the set up of different mobility patterns, permits accurate positioning and automates mote reprogramming and data downloading via the use of USB docking stations.

We believe that TrainSense can help in fostering the study of mobility in sensor network applications. In particular, TrainSense fits well the research areas in controlled mobility, which include important scenarios such as vehicular networks, barrier coverage and pipe-line monitoring. TrainSense can also help in the systematic study of sensor networks, e.g., nodes can be moved sequentially and repeatably to various locations to identify the deployment or mobility pattern that maximizes the metric of interest.

Model trains are certainly not a panacea to add mobility to WSNs. Free robots provide a quicker setup time and more path freedom. But by design, a model train setup guarantees the reproducibility of the deployment and the repeatability of the positioning and traffic patterns. A train testbed can also be installed as a permanent setup at various heights, saving space and allowing the effect of height on the RF propagation to be measured. TrainSense is an ongoing effort and there are several items planned for the near future: a modular infrastructure to allow more complex experiments, a GUI to simplify the design of mobility patterns and opening TrainSense to the community.

References

[1] http://spazioinwind.libero.it/scorzoni/motorola.htm
[2] http://www.irobot.com/
[3] http://www.mobilerobots.com/researchrobots/researchrobots/p3at.aspx
[4] Basagni, S., Carosi, A., Petrioli, C.: Controlled vs. uncontrolled mobility in wireless sensor networks: Some performance insights. In: IEEE VTC (2007)
[5] Boano, C.A., Voigt, T., Noda, C., Römer, K., Zuniga, M.: Jamlab: Augmenting sensornet testbeds with realistic and controlled interference generation. In: IPSN 2011 (2011)
[6] De, P., Raniwala, A., Krishnan, R., Tatavarthi, K., Modi, J., Syed, N.A., Sharma, S., Chiueh, T.: Mint-m: An autonomous mobile wireless experimentation platform. In: MobiSys 2006 (2006)
[7] des Roziers, C.B., Chelius, G., Ducrocq, T., Fleury, E., Fraboulet, A., Gallais, A., Mitton, N., Noel, T., Valentin, E., Vandaele, J.: Two demos using senslab: Very large scale open wsn testbed. In: DCOSS 2011(2011)
[8] Evans, L.H.D.: Localization for mobile sensor networks. In: MobiCom 2004 (2004)
[9] Förster, A., Förster, A., Leidi, T., Garg, K., Puccinelli, D., Frederick Ducatelle, S.G., Gambardella, L.M.: Motel: Towards flexible mobile wireless sensor network testbeds. In: EWSN 2011(2011)

[10] Fotouhi, H., Zuniga, M., Alves, M., Koubaa, A., Marrón, P.: Smart-hop: A reliable handoff mechanism for mobile wireless sensor networks. In: Wireless Sensor Networks, pp. 131–146 (2012)
[11] Gandham, S., Dawande, M., Prakash, R., Venkatesan, S.: Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In: IEEE GLOBECOM 2003 (2003)
[12] Grossglauser, M., Tse, D.N.C.: Mobility increases the capacity of ad hoc wireless networks. IEEE/ACM Trans. Netw. 10(4) (2002)
[13] Gu, Y., Bozda, D., Brewer, R.W., Ekici, E.: Data harvesting with mobile elements in wireless sensor networks. Comput. Netw. 50(17) (2006)
[14] Johnson, D., Stack, T., Fish, R., Flickinger, D., Ricci, R., Lepreau, J.: Truemobile: A mobile robotic wireless and sensor network testbed. University of Utah Flux Group Technical Note FTN-2005-02 April 8 (2005)
[15] Kansal, A., Rahimi, M., Estrin, D., Kaiser, W., Pottie, G., Srivastava, M.: Controlled mobility for sustainable wireless sensor networks. In: IEEE SECON (2004)
[16] Ko, A., Lau, H.Y.K.: Robot assisted emergency search and rescue system with a wireless sensor network. Int. Jour. of Advanced Science and Technology 3 (2009)
[17] Kotay, K., Peterson, R., Rus, D.: Experiments with robots and sensor networks for mapping and navigation. In: FSRR 2005 (2005)
[18] Liang, W., Luo, J., Xu, X.: Prolonging network lifetime via a controlled mobile sink in wireless sensor networks. In: IEEE GLOBECOM 2010 (2010)
[19] Liu, B., Brass, P., Douss, O., Nain, P., Towsley, D.: Mobility improves coverage of sensor networks. In: MobiHoc 2005 (2005)
[20] Luo, J., Hubaux, J.-P.: Joint mobility and routing for lifetime elongation in wireless sensor networks. In: IEEE INFOCOM 2005 (2005)
[21] Luo, J., Panchard, J., Piórkowski, M., Grossglauser, M., Pierre Hubaux, J.: Mobiroute: Routing towards a mobile sink for improving lifetime in sensor networks. In: IEEE/ACM DCOSS 2006 (2006)
[22] Rensfelt, O., Hermans, F., Gunningberg, P., Larzon, L.-Å., Björnemo, E.: Repeatable experiments with mobile nodes in a relocatable wsn testbed. Computer Journal 54(12), 1973–1986 (2011)
[23] Sibley, G., Rahimi, M.H., Sukhatme, G.S.: Robomote: A tiny mobile robot platform for large-scale ad-hoc sensor networks. In: ICRA, pp. 1143–1148 (2002)
[24] Somasundara, A., Kansal, A., Jea, D., Estrin, D., Srivastava, M.: Controllably mobile infrastructure for low energy embedded networks. IEEE Transactions on Mobile Computing 5(8), 958–973 (2006)
[25] Sugihara, R., Gupta, R.K.: Path planning of data mules in sensor networks. ACM Trans. Sen. Netw. 8(1), 1–27 (2011)
[26] Wang, G., Cao, G., Berman, P., Porta, T.F.L.: Bidding protocols for deploying mobile sensors. IEEE Trans. Mob. Comput. 6(5), 563–576 (2007)
[27] Wang, Z., Basagni, S., Melachrinoudis, E., Petrioli, C.: Exploiting sink mobility for maximizing sensor networks lifetime. In: HICSS 2005 (2005)
[28] Ye, W., Heidemann, J., Estrin, D.: An energy-efficient mac protocol for wireless sensor networks. In: IEEE INFOCOM 2002, vol. 3 (2002)

Access Control in Multi-party Wireless Sensor Networks

Jef Maerien, Sam Michiels, Christophe Huygens,
Danny Hughes, and Wouter Joosen

iMinds-DistriNet, KU Leuven, 3001 Leuven, Belgium
firstname.lastname@cs.kuleuven.be

Abstract. Emerging real world WSNs seldom exist as single owner, single application, isolated networks, but instead comprise of sensor nodes owned by multiple parties. These sensors offer multiple services to users locally or across the Internet, and travel between multiple WSNs. However, users should only have access to a limited subset of services. Due to a need for direct interactions of users with nodes, authentication and authorisation at the node level is critical. This paper presents an access control infrastructure consisting of three parts: 1) an authentication protocol to ensure authenticity of messages, 2) a role based authorisation framework to perform access control, and 3) a user management service to enable user and permission management. A prototype implementation on the ContikiOS demonstrates the validity and feasibility of node local role based access control on low power micro-controllers.

Keywords: Security Middleware, Wireless Sensor Networks, Authentication, Authorisation, Role Based Access Control.

1 Introduction

Many WSN use cases have been proposed in a wide array of application domains such as agriculture [1], logistics [2] and domotics [3]. While most first generation research focused on single application, single owner networks, it is clear that future sensing infrastructures consist of platforms owned by different parties. These nodes run multiple concurrent services[4,5] and can travel between multiple wireless networks, owned by other parties. These third parties provide Internet connectivity, giving users both local and remote access to node services. These users also want to deploy new configurations and applications onto these nodes together with party specific access policies [6].

This paper defines a node service as a software component which enables a set of related operations to be performed on the node. Users communicate with the service by sending request messages, indicating what service and operation they want to use, and receiving reply messages. Typical services in WSNs include a temperature service, offering information on current and past temperatures, or a lock actuator service, allowing lock opening.

In the described multi-user setting, strong access control is critical in order to ensure the integrity and viability of the system. A minimal access control

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 34–49, 2013.

infrastructure is comprised of three parts: 1) authentication: only known users can access services, 2) authorisation: users only have access to allowed services, and 3) management of users and permissions. Likely each node will have unique access control policies, since most nodes will be used by different parties, each with their own policies regarding application access.

Access control cannot be done at the gateway because, in the local case, the gateway is usually not part of the communication flow. In the remote case, the gateway is not necessarily owned by the same party as the sensor node [6]. The platform owners back-end could perform the necessary access control, although it would introduce significant overhead. The platform owner must then collect all access policies from all parties who currently have some application or configuration installed on the nodes for all their potential users. In some cases it would even be impossible to gather and effectively enforce such policies, for example when no back-end network connection is available.

It is thus required that this access control infrastructure exists at the node level to allow the sensor node to offer its services to users both local in the network, for example customs officer opening a container, or remote across the Internet, for example a logistics provider tracking his container [7]. Current research has investigated access control in WSNs, but often only considers authentication, with limited attention to authorisation and policy management.

The contribution of this paper is an integrated WSN access control infrastructure that allows multiple parties to securely use shared sensor node services running on top of low power micro-controller class devices and manage their access rights in a party specific fashion, which has not yet been tackled in literature. It consists of three parts: (1) an authentication protocol to ensure authenticity and confidentiality of user service request, (2) an authorisation framework using role based access control to ensure only authorised users can access services, and (3) a user management service. An implementation and evaluation of this infrastructure on the Contiki operating system and LooCI middleware demonstrates the validity of the approach.

The remainder of this paper is structured as follows. Section 2 presents a use case in the field of logistics and derives security requirements. Section 3 discusses current related work. Section 4 proposes the access control infrastructure. Section 5 provides details of the implementation and reports on the performance and security evaluation of the system, and finally Section 6 lists some promising avenues for future work and concludes this paper.

2 Use Case

2.1 Logistics Use Case

This section presents a logistics use case to illustrate the requirements as shown in figure 1. In the logistics domain, Logistics Providers (LPs) provide end to end transport to cargo owners using containers. In order to do this, the LPs use the trucks or ships of transport providers to carry containers across the world. Such trucks can drive significant distances and cross borders, requiring customs

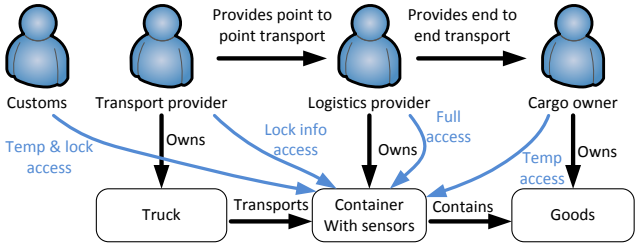


Fig. 1. Overview of the logistics use case

inspection by governments. To provide end to end transport, a logistics provider will use the services of multiple transport providers.

In order to maintain up-to-date information on and control over the conditions of the cargo, the LP equips his containers with nodes that monitor and regulate the conditions inside the container, for example temperature sensors and lock actuators. With this infrastructure, the LP can guarantee that the agreed service level, such as temperature of containers, is maintained by actively and remotely monitoring and regulating container conditions. Other parties closer to the container also want to use the services of these nodes: the transport providers want to ensure that the container remains locked, cargo owners want to validate the transport, and customs officers require that transport happens in compliance with regulations and also need to open the door to inspect the container.

This use case clearly shows that there are multiple parties in the logistics ecosystem, both local in the network, and remotely across the Internet, who want to interact with the sensor node, yet not all of them should have access to the same services. Cargo owners should not open the door, customs officers should not change the cooler settings, and so on.

2.2 Attacker Model

This paper considers two types of attackers : Network Attackers (NAs) and Physical Attackers (PAs). NAs are attackers with access to the messages exchanged over the network, in accordance with the Dolev-Yao [8] model. An NA can manipulate, duplicate or create messages, but he cannot break the underlying cryptographic primitives. He aims to gain information on or some kind of control over the network. PAs gain access to the information of the node through physical sensor probing. Standard sensor nodes are incapable of resisting PAs, unless the node is physically secure, by being either encapsulated in tamper proof hardware, or in a physically secure environment.

2.3 Security Requirements

From the preceding use case, this section derives a set of functional and security requirements for the access control infrastructure:

- **Light weight:** sensor nodes are low cost entities with limited processing, energy, and communication resources.
- **Multi-user:** multiple users have to be able to interact with the system.
- **Fine grained access control:** all users should not be able to access all services. Service access depends on the service owner and the service user.
- **Per-node management:** sensor nodes are unique entities and should not be seen as a single homogeneous network of sensor nodes. The lock node and the environment control node should not have the same access policies.
- **Protection against network attacks:** A networked attacker cannot gain access to restricted node services (authorised access only), nor can he modify messages undetected (request integrity).
- **Recovery from physical attacks:** physical attackers can gain access to all security information on the sensor nodes. When this happens, the remainder of the network should be able to recover to normal operation.

3 Related Work

Many access control systems have been proposed for WSNs. However, most systems only focus on authentication, neglecting authorisation and policy management. Current authentication systems can be divided into two categories: symmetric key and public key systems. Table 1 shows a feature overview. Symmetric key approaches offer light weight authorisation but have issues with multi-user networks and management. Public key approaches provide authentication and sometimes authorisation of user requests, but do so at a very high cost.

Table 1. Comparison of related work on authentication in WSNs with regards to requirements

	Symmetric key	public key
Light weight	+	–
Multi-user	-	+
Fine grained access control	-	+
Per node management	-	+
Resistant to node capture	-	+

The first category is symmetric key authentication. Many algorithms provide authentic broadcast using symmetric keys. Some examples are uTesla [9], TinySec [10], and sAQF [11]. TinySec uses a short Message Authentication Code (MAC) to authenticate messages using a shared secret key. UTesla uses key chains and delayed key disclosure to ensure authenticity of broadcasts. The sAQF protocol uses key rings in order to perform authentication. With each key of his ring, the user calculates a one bit MAC. Each node has a subset of these keys which it uses to verify message authenticity. These protocols often assume homogeneous networks and do not support multiple users. Furthermore none of

these protocols perform any form of authorisation nor policy management and, assuming group distribution of keys, are vulnerable to node capture.

The second category are public key authentication schemes. Messages are authenticated using the private key of the transmitter. Different protocols propose different ways to authenticate the public keys. Most propose a variation of a certificate authority, such as the work of Benenson et al [12]. Ren et al [13] use bloom filters to perform validity checking. He et al [14] propose Priccess, which uses ECC ring signatures to authenticate and authorise users. Yu et al [15] perform access control by encrypting the data with symmetric keys and encrypting these keys with public keys. Only members of a group with the necessary access rights can then decrypt the data using their private keys. This class is better protected against node capture, but does so at significant cost due to asymmetric encryption, This makes them unsuitable for low power micro-controllers. These schemes mention user revocation and key management, and some provide authorisation. However, non of these solutions offer a light weight access control framework allowing fine-grained access policies at the node level.

4 Access Control Infrastructure

This section describes the high level architecture of the access control infrastructure. First this section details the network architecture and the assumptions. Next it looks at the 3 different parts of the access control infrastructure: (1) the authentication protocol, (2) the authorisation framework, and (3) the user management service. Figure 2 shows the components added to the node system: the authentication interceptor, the authorisation layer with a proxy for each service, and the user management service which is comprised of a marshaller, a service execution component and a user database.

4.1 Network Architecture

The system identifies three entities in the network as shown in figure 3: the user platform, the Platform Owners (PO) back-end system, and the node. This paper makes a distinction between users and parties: users are individuals who use node services offered by parties. A party is an administrative entity that groups multiple users together and owns the sensor nodes. Parties offer their nodes' services to their own members and members of other parties. Each user belongs to one party, but can have permission to use the services of other parties.

The user platform is the system from which the user sends his messages to the node platform. This can be a computer, smart phone or sensor node. The user must be trusted by the PO. This trust is usually expressed in a contract between the PO and user's party. This contract states which access rights the users of a party receive in exchange for monetary compensation.

The Platform Owner (PO) is the party that owns the node. In the logistics case, this would be the Logistics Provider. The PO offers a back-end system which allows users to request access to specific node. When a user wants to

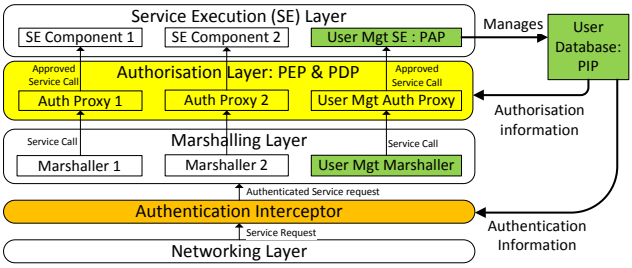


Fig. 2. Node architecture overview with three added parts: authentication interceptor (orange), authorisation framework (yellow), user management service (green)

access a node, it securely contacts the PO using Internet security protocols (SSL/HTTPS), using mutual authentication to verify the trust relationship between PO and user. A user can retrieve the PO of a node in several ways: 1) he can access an unprotected node service, which returns the PO's identity, 2) he can know ahead of time whose nodes are present, or 3) he can access a network information service and query which nodes and parties are currently in the network.

When the user knows which nodes he wants to use and who owns them, he sends a request containing the node(s) he wants to use (step 1 in figure 3). This request contains his partyId, and what access rights he wants. If permitted, he'll receive a permission token (step 2). This token is valid for a configurable amount of time ranging from minutes to weeks. When the user wants to start using node services, he must first register himself with the node by sending the token (step 3). Once registered, he can use the node services using his provided userId and key (step 4). The user can use the services until he is removed. The PO must declare in his back-end which users and parties are allowed which permissions in human-readable back-end policies. The node access control infrastructure is agnostic of how these permissions are expressed: they can be expressed in any formal policy language, such as for example XACML [16].

The final part of the network architecture is the node. The node provides certain services which can be accessed by users of the different parties. These services can provide information, reconfiguration or actuation operations. Users of parties which are sufficiently trusted are also able to add new users and manage their access rights by using the user management service. The exact access rights of users are expressed in the contract between party and PO.

4.2 Assumptions

The access control infrastructure makes the following assumptions: (1) a secure network layer is present, allowing communication through asynchronous message passing, (2) the node services are separated into a marshaller and service execution component, and (3) each node starts at its owner, where it is securely equipped with the necessary key material and security middleware.

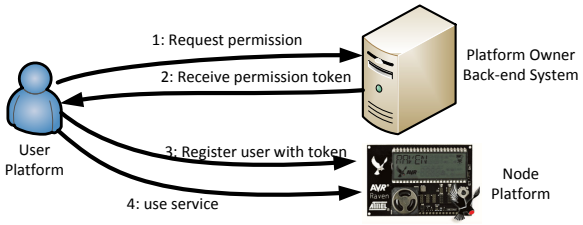


Fig. 3. User registration process

This paper assumes that the WSN uses asynchronous message passing, routed over a secure network layer. This paper is agnostic with regards to the network layer, but requires messages to consist of a payload and headers. The payload contains the functional information, the headers contain non-functional information such as security meta-data.

The protocol requires a separation of any service component into two subcomponents: the marshaller and the Service Execution Component (SEC). The marshaller interprets incoming service requests, accesses the SEC, and serializes the reply. The SEC contains the actual data and logic to perform the requests. This division is common in RPC architectures such as CORBA and RMI.

The last assumption states that the PO can securely install some initial key material onto the node. Without this initial trust setup, no new trust relationship can be deployed. This key material takes the form of a unique symmetric key shared between the PO and the specific node.

4.3 Authentication Protocol

The protocol uses a simple authentication mechanism. Each user is identified by the sensor node using a node specific numeric identifier. The user authenticates and secures his commands using a symmetric encryption algorithm in Counter with Cipher block chaining-MAC (CCM) mode. This allows for the encryption of the message payload with proof of authenticity and integrity of the payload using a Message Authentication Code (MAC). The protocol is agnostic to the actual protocol used. The actual implementation of the protocol uses AES128 due to security, resource requirements and standardisation considerations. The userId and MAC are added to the message as headers. An authenticated timestamp is optionally added to ensure message freshness.

When a node receives a service request, it is intercepted by the Authentication Interceptor (AI) before being delivered to the marshaller. The AI retrieves the userId and MAC from the message headers. It retrieves user information from the user database (Policy Information Point: PIP using XACML terminology [16]). If the userId is known, the AI decrypts and verifies the payload. If validation fails, either because of incorrect MAC or unknown userId, the message is dropped. On success, the request is delivered to the marshaller. When the marshaller sends

a reply, it must reattach the userId. The AI intercepts this reply and encrypts and authenticates the payload.

4.4 Authorisation Framework

To authorise a user, an authorisation proxy is inserted between the marshaller, and the Service Execution Component (SEC). This proxy offers the same interface as the SEC, thus can be inserted transparently with minimal effort. The proxy acts as the Policy Enforcement Point (PEP) and Policy Decision Point (PDP). It uses role based access control to authorise a user. The proxy knows the required role that a user must have. This hard-coded role requirement allows for efficient evaluation of access rights, yet is less flexible. The proxy retrieves the user's current roles from the small user database on the node, and verifies whether the user's role is sufficient.

A user can have two types of roles: node roles and party roles. A node role defines the access permission that the user has across the entire node. This allows the PO to compactly declare that a user can view or reconfigure any configuration or service on the node or to perform certain node-wide reconfigurations. A party role defines the role that the user has with that party and is only relevant for applications and configurations owned by that party.

The framework currently distinguishes 5 different access roles, listed in hierarchical order: 1) **no access**: no access to any service, 2) **viewer**: viewing information and configuration, 3) **user**: modifying existing configurations, 4)**manager**: creating and removing configuration, and 5) **administrator**: user management. Currently a higher roles also assumes all access rights of the lower roles. The number of roles and allocation of access rights is a generic framework. Roles and rights can easily be modified to adhere to domain specific requirements.

A user can have one node role, and a role for each party installed on the node. When a user adds an application or configuration, it must be assigned to a party. Either the user's default party is used, or the user specifies on behalf of which party he performs the creation operation. Of course the user needs the necessary service permission in order to create new configurations. Further service request regarding the added application will require that the requesting user has the necessary role to either that party, or node wide.

The authorisation proxy also allows for monitoring the behaviour of users of the system. Monitoring node users allows for 1) detecting potential intrusion attempts, and 2) logging of sensor node usage caused by the different users, allowing chargeback of node usage to the users.

An example from logistics as show on table 2: the Logistics Provider (LP) provides the node in the container with a lock application. The LP user has administrator rights to the node and thus this service, allowing him to open, close and manage it. A customs officer has party user permissions to services of the LP. The officer is allowed to use the lock service, namely open and close the lock and view status, but is not allowed to manage which other parties can have access. The Transport Provider(TP) user finally has only party viewer permission. He can only view current lock status.

Table 2. Overview of the permissions of the different users with the different parties and their applications

	Logistics Provider user	Transport Provider user	Customs Officer
Lock app : LP	node admin	party viewer	party user
Location app : TP	node admin	party admin	party viewer

Suppose the TP installs a localisation component on the node, which queries the truck and broadcasts it to all users registered on the node. The TP user is party administrator, and can administer which parties are allowed to view the location feed, as agreed upon in the contract that the TP and LP need to have signed beforehand. The LP user is a node administrator, so he can also administer who can view the feed on the node, such as cargo owners or customs officers. The customs party is allowed to use services offered by the LP, but only has view rights to applications offered by the TP.

While the current prototype only offers access control based on party roles, a small addition to the system would allow additional attribute based security, enabling parties only rights to certain allowed ranges of applications, even while having node viewer permissions.

4.5 User Management Service

User management is done by the User Management Service (UMS), making it the Policy Administration Point (PAP). The provided methods are: adding and removing users, adding and removing permissions, and updating key material.

The UMS provides two ways to add users, by command or by token. A service method allows users to add new users by command if he has the necessary role. To add a user by token, the user requests PO's back-end to generate a token on his behalf. The user can then send it to the UMS. This token contains following parameters: userId, partyId, node role, party role, key material, timestamp and timeout. The timestamp and timeout of the token ensure that a token will only be valid for a limited amount of time, allowing sensor node recovery. This time is configurable, allowing for tokens with a longer validity. These tokens can be requested ahead of time, and deployed in disconnected networks. The generated token is encrypted with the node's secret key, shared between the PO and the node. The node does not need to contact back-end infrastructure to ensure token validity. Once a user is added by command or token, the user only needs his userId and key to query node services.

Other provided services include a user revocation service, a role management service, and a key management service. The user revocation service allows users with a node administrator role to remove any user on the node, including themselves. Party administrators are allowed to delete users belonging to their party. The role management service allows users with a party administrator role to add and remove roles of that party from other users. This allows parties to manage access rights of users to their own services, while running on the PO's platform.

The key management service allows users to refresh keys after a certain time interval. However this does not ensure forward key secrecy. If a user's key is breached, the attacker can intercept the re-keying message and retrieve the new key. If this is detected, the user has to be removed. The user can be reinstalled by a node administrator or by using a new token.

If a user needs to manage multiple nodes, the same symmetric key can be used to allow group reconfiguration. This reduces the amount of messages the user has to send to perform reconfiguration, but at the cost of security. In case of a node breach, the user's key would be revealed and could then be used to access all other nodes on which the user used the same key.

5 Implementation and Evaluation

The protocol is implemented on top of AVR Ravens [17] running the ContikiOS [18] with IPv6 and the LooCI component middleware. The access control infrastructure can in theory be implemented using other operating systems, such as TinyOS, or other service protocols, such as CoAP. This section details the prototype, performs an evaluation, compares these values to related work and ends with a threat analysis of the proposed infrastructure.

5.1 LooCI Component Middleware

The prototype of the proposed security middleware secures the management of the LooCI component middleware [19]. Both component middleware and access control infrastructure prototype are available at code.google.com/p/looci/. LooCI is a component-based middleware existing out of an execution environment, a component model and an event-based binding model. The LooCI middleware currently supports various platforms including Contiki on AVR Ravens, SunSPOT nodes and the OSGi component model. LooCI consists out of the following parts: 1) The networking layer, 2) the event manager, 3) the component runtime, 4) the management service that is composed of a marshaller, a service execution component and a user database, and 5) the deployment module. The event manager and component runtime function as marshalling layer: the event manager dispatches events to the correct marshalling component depending on event type. The choice of LooCI was made because it supports dynamic application deployment, event based management by multiple users, an event format with headers and a clean separation between network layer, marshalling layer and service layer allowing for easy integration of access control.

LooCI uses events as the sole mechanism to communicate between components. LooCI events consist of the following parts : (1) sender information, (2) extension headers, (3) event type, and (4) event payload. The event type is a 16 bit value, which identifies what type of content the event carries. Reconfiguration and inspection of the event manager is done by the contacting the management service using management events. Deployment of new applications uses an optimised deployment protocol.

5.2 Implementation Details

The prototype intercepts events between the networking layer and the event manager. The userId and MAC are retrieved from the event headers. A 16 bit userId provides a sufficient number of users while ensuring limited overhead. The prototype authenticates and encrypts the events using AES128 in CCM mode with an 8 byte MAC, which provides a sufficient level of security with limited message overhead. The prototype authenticates the sender and receiver IP and timestamp by using the associated data field of CCM.

In order to authorise an event, an addition was made to the LooCI middleware: when an event is dispatched to a marshaller, the dispatching user is recorded by the authorisation middleware. When a marshaller calls any proxy protected service execution component, the authorisation proxy intercepts the call and verifies that the requesting user has sufficient permissions to access the requested service. If the user is authorised, the proxy calls the management SEC. If not, an error code is returned to the reconfiguration manager. When the marshaller sends out a reply, the middleware attaches the userId.

The user management service is implemented as a LooCI component, existing of a marshaller, which interprets received messages, and a SEC, which does the actual user management and contains the user data and access policies. The SEC thus performs both the functions of PIP and PAP in order to reduce implementation size.

5.3 Evaluation of Implementation

A prototype implementation of the protocol was made for ContikiOS [18] on the AVR Raven [17] running LooCI [19]. The AVR Raven is a wireless sensor node with 128kB ROM, 16kB RAM, 20MHz MCU and sleep energy usage of about 1µW. This classifies the device as a low power micro-controller. The tests evaluate the prototype against 6 metrics: message overhead, user registration overhead, message processing time, user installation time, RAM and ROM overhead, and energy cost. These figures are compared to a symmetric key approach (TinySec [10]) and a public key based approach (authenticated querying by Benenson et al. [12]) as shown in table 3. These systems were chosen because they were implemented on a sensor node and listed performance figures.

Message overhead: each message is authenticated using AES128-CCM with 8B MAC. The MAC is attached to the message in a secure payload header. The resulting message overhead is comprised of: 1) the userId header: 2B, 2) optional timestamp headers: 4B and 3) the security payload header: 8B. Each header has an additional overhead of 2B. Total: 14B-20B. To compare, TinySec message overhead is 8B, due to smaller MAC (4B). Authenticated broadcast message overhead is 20B.

User registration overhead: a user token consists of the following fields: userId, partyId, nodeRole, partyRole, userKey, timeStamp, timeOut, and MAC. The token has a total message size of 36B. The token is sent as a special user, requiring another 4B. The networking and link layer overhead of sending messages is estimated at another 32B. Total transmitted message: 72B. TinySec does

Table 3. Comparison of implementation overhead between the proposed access control infrastructure, TinySec and authenticated broadcast

Comparison criteria	Proposed protocol	TinySec	Auth. querying
Message overhead (bytes)	14	8	20
User reg overhead(bytes)	72	Not applicable	114
Message processing time(ms)	4.64	1.52	Info not available
User installation time(ms)	4.64	Not applicable	440 000
ROM overhead (bytes)	12 147	7 148	45 500
RAM overhead (bytes)	438	728	2 000

not mention any management of users. The authenticated querying overhead of deploying a user certificate is 114B.

Message processing time: the authentication of a message comprises of a) checking the userId, b) retrieving the user information, and c) decrypting and verifying the authenticity of the message. The most expensive operation is decryption: 4.64ms for a 32B block. All other operations are negligible, taking only several nanoseconds. Authorisation is much faster: only ca 20ns and comprises of a) a proxy interception, b) retrieving user access rights, and c) verifying access rights. In comparison, TinySec evaluated two block ciphers: to encrypt 32B: RC5: 1.04ms, SkipJack: 1.52ms. No data is available on the message processing overhead of Authenticated querying.

User installation time: the installation of a new user using a token requires the decryption of a 32B token, which is again the most expensive operation requiring 4.64ms. TinySec does not provide user or key management. Authenticated querying has a verification overhead of 440s to install a new certificate, mainly due to two ECC operations.

RAM and ROM overhead: table 4 lists the ROM and RAM requirements of the different parts of the implementation. The implementation contains 4 parts: 1) the AES128-CCM encryption algorithm, 2) the authentication interceptor, 3) the authorisation proxy, and 4) the user management service. The total platform has an overhead of 12 147B of ROM and 438B of RAM, which is comparable to related work: TinySec has an overhead of 7 148B of ROM and 728B of RAM. Authenticated querying has an overhead of 45.5kB of ROM and 2kB of RAM. Note that 39% of ROM and 46% of RAM usage are due to the encryption algorithms, which is shared between all security protocols.

Energy cost: this paper provides a theoretical approximation of energy usage based on device specification spreadsheets [17]. The estimated transmission energy overhead is 1.6µJ per sent or received byte, and 40 µJ per millisecond of execution time. This gives the following energy overheads: (1) **Token user installation**: transmission of 72B token and one 32B decryption operation: 1.6*72 + 4.64*40 = 300.8 µJ. (2) **Processing a secured message**: 14B transmission overhead and one decryption of 32B: 1.6*14 + 4.64*40 = 208 µJ. These figures are not entered into the table due to lack of comparable data.

Table 4. Overview of the ROM and RAM overhead of the implementation

	ROM overhead (bytes)	RAM overhead (bytes)
1 ContikiOS	42 688	9 712
2 LooCI component middleware	24 942	2 644
3 Encryption : AES128-CCM	4 746	200
4 Authentication Interceptor	1 557	129
5 Authorisation proxy	3 296	0
6 User Management Service	2 548	109
7 Total for proposed system (3-6)	12 147	438

While the overhead of the access control system is significant, it is still within the limits of a memory constrained device. This shows the feasibility of multi-user management of wireless sensor nodes, allowing direct multi-user access.

5.4 Threat Analysis

This section discusses the threats posed by Networked Attackers (NAs) and Physical Attackers (PAs). It also briefly discusses privacy threats to data.

Authorised access is ensured because NAs cannot send authentic reconfiguration requests to a protected service since that requires that the attacker can create a valid MAC. This paper assumes that the only way to forge such a MAC is by breaking the secret key, which with a sufficiently strong algorithm should be computationally impossible. **Integrity of messages** is ensured because NAs cannot modify messages which are afterwards accepted by the access control infrastructure, since he cannot create a valid MAC. **Freshness of messages** is ensure by the authenticated timestamp attached to the service request, and by the timestamp and timeout value in the user deployment token. The infrastructure thus ensures that an NA cannot access protected services nor interfere with requests other then preventing delivery.

An NA can perform a **denial of service attack**: jamming the networking layer, eliminating the possibility to send data to the node and taxing its resources. This paper does not directly address such attacks. However the system can be augmented with the following mitigation strategies: 1) temporarily disable incoming communication when an invalid message is received, preventing further resource usage, and 2) sending a notification to the PO when one or more invalid messages are detected, signalling a configuration error or attack and allowing the PO to initiate other actions.

Physical attackers can perform a **node capture attack**, revealing all key material from that node. Preventing this attack requires either costly protected hardware modules, or the physical security of the sensor nodes. When these countermeasure are not or cannot be used, a physical attacker can probe any node and gain complete access to all node information. Using the proposed framework he has no access to any other nodes due to the fact that all keys are node unique. However, if a user uses the same key for multiple nodes to allow

for group reconfiguration, then the other nodes are vulnerable too. This attack however can be recovered from or prevented. Since each user can be deleted by an administrator user, which can always be added using the node's secret key, the node can recover from a user whose key material has been compromised. Since only the long term secret key of the captured node is known, the PO can always create a new administrator on his nodes and remove the attacking user. The node secret key can only be changed with the secret key, so only the PO can perform a node re-keying. The attack can be prevented by prohibiting group keys: the user has to use a unique key for accessing each node. While this increases overhead, it significantly increases the security of the platforms. It is thus advisable to use this policy for critical node services.

Due to the ubiquitous nature of sensor nodes, it is possible that in certain contexts, the sensors pick up **privacy** sensitive information. It is possible that in such cases, the PO should not perform certain operations or readings because they intrude on the privacy of the tracked subject. However, such attacks are non-technical and concern the legal issues of sensing by POs rather than technical ones. In such cases, these constraints will be entered in the contract that exists between the users and the POs, as stated in section 4.1. The contract will then state that the PO is not allowed to read certain data, which can potentially be translated in user permissions and deployed on the node.

6 Future Work and Conclusion

6.1 Future Work

This paper presented a light weight access control system allowing multiple users from different parties to securely share WSN services. However the current architecture has still some possible avenues of future work.

The first avenue is the exploration of the boundaries of access control policy expressiveness. The current implementation allows for basic role based access control. Access to a service only depends on the user identity and his roles. More advanced ways to perform access control will be investigated where the decision also depends on the arguments of the service call in addition to the role of the requester. For example when performing reconfigurations, it is desirable to be able to restrict the range of values which a user might set, such as the sampling rate of a temperature component.

Secondly, in order to securely share sensor nodes between multiple parties, the execution of requests and configurations of the different parties must be isolated in order to prevent interference. This requires hardware support from the node platforms, which the current generation lacks. Future work could investigate new hardware designs allowing isolated execution of code on sensor nodes.

The third point of future work is the monitoring and auditing of user behaviour. Once a strong authentication and authorisation system is set up, the node can log and audit user behaviour. This auditing can be used to allow charge-back of node usage, enabling POs to recuperate the cost of deploying a wireless sensor network, and monitoring behaviour for malicious use.

6.2 Conclusion

Future sensor networks will operate in dynamic multi-user environments. Multiple users will interact with low resource nodes to retrieve information or modify node configuration. Due to strong resource constraints and unique interaction patterns, traditional access control systems cannot be used. Contemporary research in this area often neglects providing authorisation functionality or adequate management of access control policies.

This paper presented an access control infrastructure for resource constrained wireless sensor nodes. The infrastructure uses a protocol based on symmetrical encryption for authentication and a role based access control framework for authorisation. It provides a management service allowing easy addition, modification and removal of users and roles. The combination of these elements creates a strong access control infrastructure for resource limited WSNs.

The infrastructure is evaluated by means of a prototype which shows the validity of the approach. The overhead of the access control mechanism is significant, but originates mostly from expensive cryptographic algorithms. These algorithms can however be shared with other node systems (secure storage or secure routing), limiting the actual increased cost of the infrastructure. This paper shows that low power micro-controllers can support secure multi-user access control of node services with per user and per party unique access policies.

Acknowledgements. Research partially funded by a Ph.D. grant of the Agency for Innovation by Science and Technology (IWT), the Research Fund KU Leuven, and the EU FP7 project NESSoS. With the financial support from the Prevention of and Fight against Crime Programme of the European Union (B-CCENTRE). Project is conducted in the context of the IWT-SBO-STADiUM project No. 80037.

References

1. Wark, T., Corke, P., Sikka, P., Klingbeil, L., Guo, Y., Crossman, C., Valencia, P., Swain, D., Bishop-Hurley, G.: Transforming agriculture through pervasive wireless sensor networks. IEEE Pervasive Computing 6(2), 50–57 (2007)
2. Zhang, Z., Chen, Q., Bergarp, T., Norman, P., Wikstrom, M., Yan, X., Zheng, L.R.: Wireless sensor networks for logistics and retail. In: INSS, June 17-19, pp. 1–4. IEEE Computer Society, Washington, DC (2009)
3. Zatout, Y., Campo, E., Llibre, J.F.: Wsn-hm: Energy-efficient wsn for home monitoring. In: ISSNIP, pp. 367–372. IEEE Computer Society, Washington, DC (2009)
4. Leontiadis, I., Efstratiou, C., Mascolo, C., Crowcroft, J.: SenShare: Transforming Sensor Networks into Multi-application Sensing Infrastructures. In: Picco, G.P., Heinzelman, W. (eds.) EWSN 2012. LNCS, vol. 7158, pp. 65–81. Springer, Heidelberg (2012)
5. Oliveira, L.B., Kansal, A., Priyantha, B., Goraczko, M., Zhao, F.: Secure-tws: Authenticating node to multi-user communication in shared sensor networks. In: IPSN 2009, pp. 289–300. IEEE Computer Society, Washington, DC (2009)

6. Huygens, C., Joosen, W.: Federated and shared use of sensor networks through security middleware. In: ITNG 2009, pp. 1005–1011. IEEE Computer Society, Washington, DC (2009)
7. Priyantha, N.B., Kansal, A., Goraczko, M., Zhao, F.: Tiny web services: design and implementation of interoperable and evolvable sensor networks. In: SenSys 2008, pp. 253–266. ACM, New York (2008)
8. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2), 198–208 (1983)
9. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: Spins: security protocols for sensor networks. Wireless Networking 8(5), 521–534 (2002)
10. Karlof, C., Sastry, N., Wagner, D.: Tinysec: a link layer security architecture for wsns. In: SenSys 2004, pp. 162–175. ACM, New York (2004)
11. Werner, F., Benenson, Z.: Formally verified authenticated query dissemination in sensor networks. In: SPECTS 2009, pp. 154–161. IEEE Press (2009)
12. Benenson, Z.: Realizing robust user authentication in sensor networks. In: Real-World Wireless Sensor Networks (REALWSN) (2005)
13. Ren, K., Lou, W., Zhang, Y.: Multi-user broadcast authentication in wireless sensor networks. In: SECON 2007, pp. 223–232 (June 2007)
14. He, D., Bu, J., Zhu, S., Chan, S., Chen, C.: Distributed access control with privacy support in wireless sensor networks. IEEE Transactions on Wireless Communications 10(10), 3472–3481 (2011)
15. Yu, S., Ren, K., Lou, W.: Fdac: Toward fine-grained distributed data access control in wireless sensor networks. IEEE Transactions on Parallel and Distributed Systems 22(4), 673–686 (2011)
16. Godik, S., Moses, T.: Extensible access control markup language (xacml), v2.0. Technical report, OASIS (2005)
17. Atmel: Avr raven Available as, http://www.atmel.com/tools/AVRRAVEN.aspx
18. Dunkels, A., Gronvall, B., Voigt, T.: Contiki - a lightweight and flexible operating system for tiny networked sensors. In: LCN 2004, pp. 455–462. IEEE Computer Society, Washington, DC (2004)
19. Hughes, D., Thoelen, K., Horré, W., Matthys, N., Cid, J.D., Michiels, S., Huygens, C., Joosen, W., Ueyama, J.: Building wsn applications with looci. IJMCMC 2(4), 38–64 (2010)

HybridStore: An Efficient Data Management System for Hybrid Flash-Based Sensor Devices

Baobing Wang and John S. Baras

Department of Electrical and Computer Engineering
The Institute for Systems Research
University of Maryland, College Park
{briankw,baras}@umd.edu

Abstract. In this paper, we propose HybridStore, a novel efficient resource-aware data management system for flash-based sensor devices to store and query sensor data streams. HybridStore has three key features. Firstly, it takes advantage of the on-board random-accessible NOR flash in current sensor platforms to guarantee that all NAND pages used by it are *fully occupied* and written in a *purely sequential* fashion, and expensive in-place updates and out-of-place writes to an existing NAND page are *completely avoided*. Thus, both raw NAND flash chips and FTL-equipped (Flash Translation Layer) flash packages can be supported efficiently. Secondly, HybridStore can process typical joint queries involving both time windows and key value ranges as selection predicate extremely efficiently, even on large-scale datasets. It organizes a data stream into segments and exploits a novel index structure that consists of the inter-segment skip list, and the in-segment β-Tree and Bloom filter of each segment. Finally, HybridStore can trivially support time-based data aging without any extra overhead because no garbage collection mechanism is needed. Our implementation and evaluation with a large-scale real-world dataset in TinyOS reveals that HybridStore can achieve remarkable performance at a small cost of constructing the index.

1 Introduction

One of the main challenges in wireless sensor networks is the storage and retrieval of sensor data. Traditional centralized data acquisition techniques (e.g., [8]) suffer from large energy consumption, as all the readings are transmitted to the sink. In long-term deployments, it is preferable to store a large number of readings *in situ* and transmit a small subset only when requested [7]. This framework becomes practically possible with the new generation NAND flash that is very energy efficient with high capacity. Recent studies show that the NAND flash is at least *two orders of magnitude cheaper* than communication and *comparable in cost* to computation [10]. Therefore, extending the NAND flash to off-the-shelf low-end sensor platforms can potentially improve in-network processing and energy-efficiency substantially.

However, due to the distinctly different read and write semantics of the NAND flash, and tightly constrained resource on sensor platforms, designing an efficient resource-aware data management system for flash-based sensor devices is a very challenging task. Existing techniques, such as LA-Tree [1], μ-Tree [5], B-File [13], FlashDB [14]

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 50–66, 2013.

and PBFilter [18], are not applicable to sensor platforms due to their large RAM footprints. Capsule [9] provides a stream-index object to store data stream efficiently, however, with very limited supports for general queries. Other works, such as TL-Tree [6] and FlashLogger [12], can only process time-based queries.

The most related works are Antelope [17] and MicroHash [7]. Antelope is a lightweight database management system for low-end sensor platforms, which enables runtime creation and deletion of databases and indexes. However, its main index for value-based queries, MaxHeap, requires expensive byte-addressable random writes in flash. Therefore, Antelope is more suitable for the NOR flash, which limits its performance because the NOR flash is much slower and more energy-consuming compared to the NAND flash. In addition, it can only retrieve *discrete values* in value-based range queries. MicroHash is an efficient index structure for NAND flash-based sensor devices, supporting value-based equality queries and time-based range queries *separately*. However, it suffers from out-of-place writes to existing pages, resulting in long chains of partially occupied pages. They alleviate this problem by combining multiple such pages into a fully occupied page, which induces extensive page reads and writes during insertions. More importantly, neither Antelope nor MicroHash can support joint queries involving both time windows and value ranges as selection predicate efficiently. That means, even though a query just wants to search readings within a certain value range in a small time window, they still needs to traverse the *whole global* index.

Existing works do not take advantage of both the on-board *random-accessible* NOR flash that is quite suitable for index structures, and external economical energy-efficient NAND flash with high-capacity, which is ideal for massive data storage. In this paper, we propose HybridStore, a novel efficient data management system for resource-constrained sensor platforms, which exploits both the on-board NOR flash and external NAND flash to store and query sensor data streams. In order to completely avoid expensive in-place updates and out-of-place writes to an existing NAND page, the index structure is created and updated in the NOR flash. To handle the problem that the capacity of the NOR flash on low-end sensor platforms is very limited (512KB to 1MB), HybridStore divides the sensor data stream into segments, the index of which can be stored in one or multiple erase blocks in the NOR flash. Since the NAND flash is much faster and more energy-efficient for reading, the index of each segment is copied to the NAND flash after it is full. Therefore, all NAND pages used by HybridStore are fully occupied and written in a purely sequential fashion, which means it can support both raw NAND flash chips and FTL-equipped flash packages efficiently.

HybridStore can process typical joint queries involving both time windows and key value ranges as selection predicate extremely efficiently even on large-scale datasets, which sharply distinguishes HybridStore from existing works. The key technique is a novel index structure that consists of the inter-segment skip list, and the in-segment β-Tree and Bloom filter of each segment. The inter-segment skip list can locate the desired segments within the time window of a query, and skip other segments efficiently. The β-Tree of a segment is the key data structure to support value-based queries. It exploits a simple prediction-based method to split each node in the tree adaptively to generate a rather balanced tree, even when key values are very unevenly distributed. The Bloom filter of a segment facilitates value-based *equality* queries inside that segment, which

can detect the existence of a given key value efficiently. Our index can eliminate a substantial number of unnecessary page reads when processing joint queries.

In addition, HybridStore can trivially support time-based data aging without any extra overhead, because no garbage collection mechanism is needed here, which will induce extensive page reads and writes to move valid pages within the reclaimed erase blocks to new locations. HybridStore can be used as a storage layer to provide a higher-level abstraction to applications that need to handle a large amount of data. For example, the design and implementation of Squirrel [11] can become much simpler if HybridStore is adopted for storage management.

The rest of this paper is organized as follows. We discuss the design considerations in Section 2. In Section 3, we explain the design of HybridStore. Our experimental results and analysis are presented in Section 4. Finally, Section 5 concludes this paper.

2 Design Considerations

In this section, we first discuss various constraints that make the design of HybridStore challenging. Then we discuss several design principles that result from these constraints.

2.1 Design Constraints

Flash Constraints Flash memory complicates the design of HybridStore by prohibiting in-place updates. Unlike magnetic disks, flash memories only allow bits to be programmed from 1 to 0. To reset a bit to 1, a large block of consecutive bytes must be erased, which is typically several kilobytes large [17]. There are two kinds of flash memories. The NOR flash is byte-addressable and permits random access I/O, but the erase blocks are very large. The NAND flash is page-oriented and limited to sequential writes within an erase block that can be significantly smaller than a NOR flash block. Reads and writes on the NAND flash happen at a page granularity. Since each page can be written only once after each complete block erasure, out-of-place writes to an existing NAND page are complex and very expensive. Portable flash packages such as SD cards and CF cards exploit a Flash Translation Layer (FTL) to hide many of these complexities and provide a disk-like interface. However, random page writes on current FTL-equipped devices are still *well over two orders of magnitude more expensive* than sequential writes, while semi-random writes are very efficient [13].

Table 1. Performance of flash memory operations

	Read		Write		Block Erase	
	Latency	Energy	Latency	Energy	Latency	Energy
Atmel NOR (per byte)	$12.12\mu s$	$0.26\mu J$	$12.6\mu s$	$4.3\mu J$	$12ms/2KB$	$648\mu J/2KB$
Toshiba NAND (per page)	$969.61\mu s$	$57.83\mu J$	$1081.42\mu s$	$73.79\mu J$	$2.6ms/16KB$	$65.54\mu J/16KB$

Energy Constraints. NOR flash and NAND flash are very different in speed and energy-efficiency. Table 1 shows the latency and energy consumption of each operation on the 512KB Atmel AT45DB041B NOR flash [3,10] equipped on the Mica family,

and the 128MB Toshiba TC58DVG02A1FT00 NAND flash [9] used extensively in the research community. Each NAND block consists of 32 pages of 512B each. We can observe that the NAND flash has a much larger storage capacity, and much faster and more energy-efficient I/O, while the only advantage of the NOR flash is random access and byte-addressable. These features influence the design of HybridStore extensively.

Memory Constraints. RAM is very limited on sensor platforms. Current low-end sensor platforms (e.g., MicaZ, Iris and Tmote Sky) are equipped with no more than 10KB RAM. Even on advanced sensor platforms (e.g., iMote2) with tens of megabytes RAM, RAM is still a very precious resource, because complex data processing applications with much higher RAM demands are expected to run on these platforms. Therefore, HybridStore must be designed to minimize the RAM footprint.

2.2 Design Principles

Given the above constraints, the design of HybridStore should follow a few design principles. Firstly, the system should take advantage of both the on-board NOR flash and external NAND flash. To support both raw NAND flash and FTL-equipped devices, random page writes should be avoided. To increase the energy-efficiency, out-of-place writes to an existing NAND page should be eliminated as well. Secondly, writes should be batched to match the write granularity of the NAND flash, which can be satisfied by using a page write buffer in RAM. In addition, since the NAND flash is much faster and more energy-efficient, most or even all reads should happen in the NAND flash. Thirdly, the system should support multiple storage allocation units and align them to erase block boundaries to minimize reclamation costs. Moreover, the system should maintain most data structures and information in flash whenever possible to minimize the RAM footprint. Finally, HybridStore should support data aging to reclaim space for new data when the NAND flash starts filling up with minimum overhead.

3 HybridStore

HybridStore provides the following interface to insert and query sensor readings:

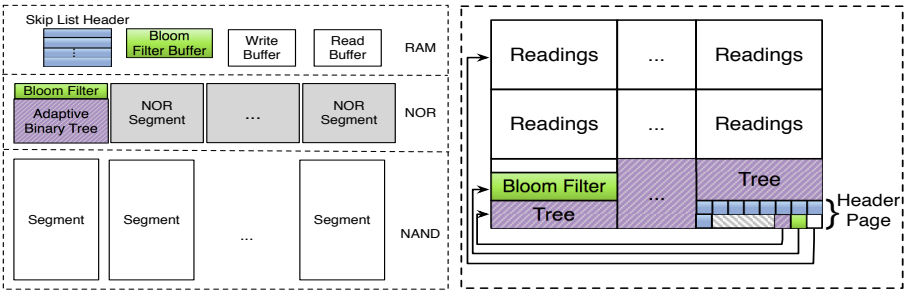
- command error_t insert(float key, void* record, uint8_t length)
- command error_t select(uint64_t t_1, uint64_t t_2, float k_1, float k_2)

The `select` function supports joint queries involving both time windows ($[t_1, t_2]$) and key ranges ($[k_1, k_2]$) as their *selection predicate*. We assume that a sensor mote generates readings periodically or semi-periodically as in adaptive sensing, and each reading can contain measurements from multiple types of sensors.

HybridStore consists of the following main components: Storage Manager, Index Manager, Query Processor, and Data Aging and Space Reclamation Module.

3.1 Storage Manager

The Storage Manager allocates storage space from the NOR flash and the NAND flash for index construction and data storage upon request. Fig. 1a shows the storage hierarchy. Both the NOR flash and the NAND flash are organized as circular arrays, resulting in the minimum RAM overhead, because we do not need to maintain a data structure in RAM to track free blocks. In addition, this organization directly addresses the write constraints, space reclamation, and wear-leveling requirements (Section 3.4).



(a) Storage hierarchy (b) NAND segment structure

Fig. 1. System architecture

At the highest level, the NOR flash is divided into equally-sized segments, each of which consists of one or multiple consecutive erase blocks. The NOR flash is allocated and reclaimed at the granularity of a segment. The NAND flash is allocated at the granularity of an erase block, but reclaimed at the granularity of a segment, which *logically* consists of several consecutive erase blocks storing readings, the index copied from the corresponding NOR segment (colored in green and purple), and the header page, as shown in Fig. 1b.

To minimize the RAM footprint of HybridStore, and comply with the write and read granularity of the NAND flash, only four absolutely necessary data structures are maintained in RAM. The write buffer is of one page size to batch the writes of readings to the NAND flash, and the read buffer is two pages large (one for index page reads and the other for data page reads). The other two data structures are the skip list header and Bloom filter buffer that are discussed in the next section.

3.2 Index Manager

In this section, we present the most important component of HybridStore: the *Index Manager*. HybridStore leverages a memory hierarchy to achieve more efficient index operations. Specifically, HybridStore divides a sensor data stream into dynamically-sized partitions, each of which is stored in a logical NAND segment. The index for this partition is first "cached" in a NOR segment, which is then copied to the corresponding logical NAND segment when it is filled. Next, HybridStore allocates a new NOR segment for the next partition, and stores its readings in a new NAND segment.

HybridStore exploits an *inter-segment skip list* to locate the segments covered by $[t_1, t_2]$ efficiently. Within each segment, HybridStore maintains an *in-segment β-Tree* to locate all readings within $[k_1, k_2]$ efficiently. To speed up the processing of value-equality queries (i.e., $k_1 = k_2$), an *in-segment Bloom filter* is also created for each segment to quickly detect the existence of a given key.

HybridStore chooses the partition-based index scheme instead of a global index as in [7,17] for the following reasons. Firstly, typical queries on sensor data always involve time windows. Since each logical NAND segment only stores the readings of a partition that corresponds to a small time window, many logical NAND segments outside the query time window can be skipped, reducing a substantial number of unnecessary page reads during query processing. Secondly, the number of readings in a partition is very limited compared to that in the whole steam. This allows index structure optimization and much cheaper index construction costs. Thirdly, the range of the key values of readings in a partition is very small. When processing a query with a value range within its selection predicate, many logical NAND segments outside the query value range can be skipped as well, further reducing many unnecessary page reads. Therefore, HybridStore is extremely efficient to process joint queries with both time windows and value ranges as their selection predicates. Finally, since all logical NAND segments are relatively independent of each other, HybridStore can support time-based data aging without any garbage collection mechanism, resulting in the substantially reduced overhead.

Now we discuss the index structure of HybridStore in details, which consists of three main modules: the inter-segment skip list, the in-segment β-Tree, and the in-segment Bloom filter. In the last subsection, the procedure to copy the "cached" index from the NOR flash to the NAND flash is described as well.

Inter-segment Skip List. The key issue to process a query is to locate the segments containing the readings within $[t_1, t_2]$. A naive approach is to scan the headers of all the segments one by one, if the time window of all the readings in a segment is available in its header and all segments are chained using previous segment address pointers. The expected cost is linear with the number of segments. However, we actually can do much better by exploiting the fact that the time windows of all segments are naturally ordered in descending order. To efficiently locate the desired segments, we organize all segments as a skip list [16]. A skip list is an ordered linked list with additional forward links added randomly, so that a search in the list can quickly skip parts of the list. The expected cost for most operations is $O(\log n)$, where n is the number of items in the list. Since segments are created in descending timestamp order, a new segment is always inserted at the front of the skip list, which can be efficiently implemented in a flash.

The skip list consists of a header node in RAM and a node in the header page of each segment (colored in blue in Fig. 1). Each node keeps *MaxLevel* number of forward pointers, each of which references the address of the header page of a segment and the timestamp of the first (oldest) reading stored in that segment. All pointers in the header node are initialized to *null* at first. When a segment is full, the skip-list node in its header page is created and inserted into the skip list before a new segment starts as follows. Firstly, a level $l \in [1, MaxLevel]$ is generated for it, randomly, such that a fraction p ($p = \frac{1}{2}$ typically) of the nodes with level i can appear in level $i + 1$. The maximum level of all segments in the current system, *curMaxLevel*, is updated if it is smaller than l.

Then every pointer in level $i \in [1, l]$ in the skip-list header, is copied as the level i pointer to the header page. Finally, the timestamp of the first reading in this segment and the header page address are written as the new level i pointer to the skip-list header.

The header page of each segment contains the timestamps of the first and the last readings stored in this segment. To search the segments containing readings within $[t_1, t_2]$, we first locate the most recent segment with a *start* timestamp smaller than t_2, using an algorithm similar to the search algorithm in [16]. The subsequent segments can be located by following the level 1 pointer in the skip-list node of each segment, until a segment with a start timestamp smaller than t_1 is encountered.

In-segment β-Tree. To support value-based equality and range queries, HybridStore exploits an adaptive binary tree structure, called β-Tree, to store the index for each segment. The β-Tree consists of a set of equally-sized buckets, each of which stores index entries within a certain value range. The header of a bucket consists of its value range, the bucket IDs of its two children, and the value to split its value range to obtain the value ranges for its children. The β-Tree is first created and updated in a NOR segment, and then copied to the corresponding logical NAND segment.

An index entry $< key, addr >$ is inserted into the β-Tree as follows. Suppose the current bucket is b and $key \in (b.min, b.max]$, then this entry is appended to b. Otherwise, we traverse the β-Tree to locate the leaf bucket b' such that $key \in (b'.min, b'.max]$, and append this entry to b'. If b (or b') is full, its value range is split into $(min, mid]$ and $(mid, max]$, and a new bucket b_{new} is allocated as its left child if $key \leq mid$, or as its right child otherwise. Then the headers of both b (or b') and b_{new} are updated correspondingly and this entry is inserted into b_{new}. Since the children of a bucket are allocated only if necessary, it is possible to have $b' = null$ in the above case if, for example, b' is the left child of its parent but only the right child of its parent has been allocated since its splitting. In this case, a new bucket is allocated for b' first.

Instead of splitting the value range evenly as in [7,17], HybridStore uses a prediction-based adaptive bucket splitting method, because readings are temporally correlated, which can be used to predict the value range of the following readings based on the most recent readings, and split a bucket range accordingly. This method is preferred for the following reasons. Firstly, each partition contains readings in a small value range. If the very large range for all possible key values is split evenly in each step, the index tree of a segment may degenerate to a long list at the beginning, resulting in more time and energy consumption to traverse the index. Secondly, although the whole range is very large, most readings will belong to a much smaller range due to their uneven distribution. As shown in Fig. 11 in [7], over 95% of the temperature measurements belong to $[30\,°\text{F}, 80\,°\text{F}]$, while the whole range is $[-60\,°\text{F}, 120\,°\text{F}]$. Again, the evenly splitting method will result in a rather unbalanced tree.

HybridStore exploits the Simple Linear Regression estimator for prediction due to its simplicity in computation, negligible constant RAM overhead, and high accuracy for temporally correlated data. HybridStore buffers the *keys* of the most recent m readings and predicts the value range for the following $2m$ readings, where m is the number of entries that can be stored in a bucket. Suppose the range of the current bucket is $(x, y]$ and the predicted range is $[l, h]$, the splitting point *mid* is computed as:

$$mid = \begin{cases} (l+h)/2 & \text{if } [l,h] \subseteq (x,y] \\ (x+y)/2 & \text{if } (x,y] \subseteq [l,h] \\ (x+h)/2 & \text{if } l \le x < h \le y \text{ and } 2m * (h-x)/(h-l) > m \\ \max(h,(x+y)/2) & \text{if } l \le x < h \le y \text{ and } 2m * (h-x)/(h-l) \le m \\ (l+y)/2 & \text{if } x \le l < y \le h \text{ and } 2m * (y-l)/(h-l) > m \\ \min(l,(x+y)/2) & \text{if } x \le l < y \le h \text{ and } 2m * (y-l)/(h-l) \le m \end{cases}$$

Compared to MaxHeap [17], an evenly splitting scheme, HybridStore can generate a more balanced tree. For example, suppose each bucket can store the index entries generated in half an hour, the current temperature is 80 °F and will increase 1 °F every half an hour, and the whole range is [−60 °F, 120 °F]. Assuming HybridStore can predict accurately, the root will be split with $mid = 82$ °F, and its right child will be split with $mid = 84$ °F. The resulting β-tree will have three layers for readings in the following 2.5 hours, while MaxHeap degenerates to a list with 5 buckets. In addition, MaxHeap allocates two child buckets at the same time when a bucket needs to be split, resulting in more wasted space with empty buckets. To handle the uneven distribution of key values, MaxHeap selects a bucket for an index entry based on the hashed key value, but store the unhashed key. As a result, MaxHeap can only retrieve *discrete values* in a range search. More importantly, however, any spatial correlations of index insertions are destroyed. After hashing, the index entries for consecutive readings are very likely to be stored in many different buckets, which will increase the read costs both for bucket locating and query processing substantially. On the contrary, these entries will be stored in the same bucket in the β-Tree. Finally, different from [7,17], our prediction-based adaptive splitting scheme does not require a priori knowledge of the whole range, which makes HybridStore more suitable for general applications.

In-segment Bloom Filter. A special case of value-based queries is value-based *equality* search that is also often desired [17]. Although β-Trees can support this kind of queries, HybridStore needs to traverse the whole β-Tree even when the given key does not exist in a segment. To better support these queries, HybridStore creates a Bloom filter [4] for each segment to detect the existence of a key value in this segment efficiently.

A Bloom Filter (BF) is a space-efficient probabilistic data structure for membership queries in a set with low false positive rate but no false negative. It uses a vector of v bits (initially all set to 0) to represent a set of n elements, and q independent hash functions, each producing an integer $\in [0, v-1]$. To insert an element a, the bits at positions $h_1(a), \ldots, h_q(a)$ in the bit vector are set to 1. Given a query for element a', all bits at positions $h_1(a'), \ldots, h_q(a')$ are checked. If any of them is 0, a' cannot exist in this set. Otherwise we assume that a' is in this set.

Since a Bloom filter requires bit-level random writes, it must be buffered in RAM. However, if a Bloom filter is used to represent all readings in a segment, this buffer size may be very large in order to keep p very low. For example, suppose a segment can store 4096 readings and three hash functions are used, in order to keep $p \approx 3.06\%$, then the size of its Bloom filter buffer must be at least 4KB.

To reduce the RAM footprint, HybridStore *horizontally* partitions the large Bloom filter of a segment into a sequence of small fix-sized Bloom filters sections, and allocates

a small buffer in RAM for a section. Suppose the buffer size is v bits, the number of hash functions is q, and the desired false positive rate is p, the maximum number n of readings that a BF section is able to represent can be calculated from the equation $p = \left(1 - \left(1 - \frac{1}{v}\right)^{qn}\right)^q$. Whenever n readings have been inserted into the current BF section, the BF buffer is flushed to the current NOR segment, and then initialized for the next section. In our implementation, $v = 2048$ bits, $q = 3$, $p = 3.06\%$, and $n = 256$.

Algorithm 1. checkBF($addr, l, k$)

Input: *addr*: start address of the BF pages, *l*: length of a BF fragment in bytes, *k*: key value
Output: *true* if there is a record with the given key in this segment; *false* otherwise

```
1:  h ← hashcode(k); bv ← createBitVector(⌈NAND_Page_Size / l⌉);
2:  for i = 0 → h.size do
3:      f ← loadPage(addr + ⌊h[i] / 8l⌋ * NAND_Page_Size); bv.setAll();
4:      for j = 0 → bv.size do
5:          mask ← 0x80 >> (h[i] % 8); offset ← h[i] % 8l;
6:          if f[j * l + ⌊offset / 8⌋] & mask == 0 then bv.clear(j); end if
7:      end for
8:      exist ← false;
9:      for i = 0 → bv.size do
10:         exist = exist | bv.get(i);
11:     end for
12:     if !exist then return false; end if
13: end for
14: return true;
```

A drawback of horizontal partitioning is that all BF sections of a segment must be scanned to decide whether the given key exists in this segment. HybridStore addresses this drawback by *vertically* splitting these BF sections into fragments and group them into pages when the NOR segment is copied to the logical NAND segment. Assume there are s BF sections in the current segment when it is full. Then the size of a fragment is $l = \left\lfloor \frac{NAND\_Page\_Size}{s} \right\rfloor$ bytes, so that the bits in the range $[i * 8l, (i + 1) * 8l - 1]$ from every BF section are grouped to page $i \in \left[0, \lceil \frac{NAND\_Page\_Size}{l} \rceil - 1\right]$. Thus HybridStore only needs to scan at most q pages at $\left\lfloor \frac{h_1(k)}{8l} \right\rfloor, \ldots, \lfloor \frac{h_q(k)}{8l} \rfloor$ when checking key k, as shown in Algorithm 1. For each hash code $h_i(k)$, HybridStore firstly loads the page containing all the $h_i(k)$-th bits of every BF section (Line 3), and then check the corresponding bit in each BF fragment (Line 4–7). If the corresponding bit is not set in any fragment in that page (Line 9–11), we can conclude that this key does not exist (Line 12). Note that a segment cannot store more than $n * NAND\_Page\_Size$ readings to guarantee $l \neq 0$.

Copy Index from the NOR Flash to the NAND Flash. Since the NAND flash is much faster and more energy-efficient, the index of a segment that is created and updated in the NOR flash is copied to the NAND flash after this segment is full as follows. Firstly, the β-Tree is copied and multiple consecutive buckets are written to the same page if they can fit in. The Query Processor is able to translate the bucket ID to the right page address and offset to read a desired bucket. Therefore, the bucket size should be $\frac{1}{2^i}$ of the NAND page size. Secondly, the BF sections are copied as described above. Finally, the time window

and value range of all readings in this segment, the addresses of the first page for readings, of the β-Tree, and of the Bloom filter, the length of a BF fragment, and the skip-list node are written to the *next* page, which is the header page of this segment. Therefore, all page writes in the NAND flash are purely sequential; the reason why HybridStore can support both raw NAND flash chips and FTL-equipped NAND flash cards efficiently.

3.3 Query Processor

Algorithm 2 describes how HybridStore can efficiently process joint queries. The basic idea is to skip all the segments that do not satisfy the selection predicate by checking their header pages, or do not contain the given key by checking their Bloom filters.

Algorithm 2. select(t_1, t_2, k_1, k_2)

Input: Time window $[t_1, t_2]$ and key value range $[k_1, k_2]$ of a query
Output: The records that satisfy the query criteria

```
 1: addr ← skipListSearch(t2);
 2: while addr ≥ 0 do
 3:     addr ← segmentSearch(addr, t1, t2, k1, k2);
 4: end while
 5: signal finished;
 6: function SEGMENTSEARCH(addr, t1, t2, k1, k2)
 7:     f ← loadPage(addr);
 8:     if [k1, k2] ∩ [f.minK, f, maxK] ≠ ∅ then
 9:         if t1 == t2 then                          ▷ Simple time-based equality query
10:             scaleBinarySearch(f.dataBList, t1);
11:             return -1;
12:         else if k1 == k2 then                     ▷ Value-based equality query
13:             if checkBF(f.bfAddr, f.bfFragSize, k1) == false then
14:                 return (f.startT > t1 && f.sl[0].time ≥ System.minT) ? f.sl[0].addr : −1;
15:             end if
16:         end if
17:         q ← createQueue(f.idxBList[0]);           ▷ Traverse β-tree
18:         while !q.empty() do
19:             b ← loadBucket(q.dequeue());
20:             for i = 0 → b.record.size do
21:                 if (b.record[i] ≠ null) && (b.record[i].key ∈ [k1, k2]) then
22:                     dataP ← loadPage(b.record[i].addr);
23:                     if dataP[b.record[i].addr % P].timestamp ∈ [t1, t2] then
24:                         signal dataP[b.record[i].addr % P];
25:                     end if
26:                 end if
27:             end for
28:             if (b.left ≠ null) && (b.middle ≥ k1) then q.enqueue(b.left); end if
29:             if (b.right ≠ null) && (b.middle < k2) then q.enqueue(b.right); end if
30:         end while
31:     end if
32:     return (f.startT > t1 && f.sl[0].time ≥ System.minT) ? f.sl[0].addr : −1;
33: end function
```

HybridStore starts by locating the most recent segment within the time window using the inter-segment skip list (Line 1), and then scans segments sequentially until the whole time window has been covered (Line 2–4). For each segment, its header page is loaded first. If this segment potentially contains readings within the value range of the query (Line 8), HybridStore begins to traverse its index. Otherwise, this segment will be skipped. Two special cases are treated separately. Time-based *equality* queries are processed using a scale binary search (Line 9) on the pages storing readings, which exploits the fact that sensor readings are generated periodically or semi-periodically to refine the middle in each iteration, similar to [7]. For value-based *equality* queries, HybridStore first checks the existence of the key in this segment using Algorithm 1 (Line 12). If this key does not exist, this segment will be skipped as well. For general joint queries, the β-Tree of this segment is traversed using the Breadth-First Search algorithm (Line 17–30).

To reduce the RAM footprint, HybridStore returns readings on a record-by-record basis. Actually, the β-Tree traversal is also implemented in a split-phase fashion (bucket-by-bucket using the *signal-post* mechanism), although a queue-based implementation is presented here for clarity. In addition, to take advantage of the temporal correlations and spatial locality of readings, a small record pool is applied here to buffer the data page addresses that will be loaded in increasing order. Therefore, instead of loading the data page immediately (Line 22), the address of each reading is translated to the corresponding page address that is then added to the pool. When the pool is full or the β-Tree traversal is finished, HybridStore loads these pages in order and scans each page to return the readings that satisfy the selection predicates.

3.4 Data Aging and Space Reclamation

As shown in [10], a sensor mote can store over 10GB data during its lifetime. If the capacity of the external NAND flash is not big enough to store all these data, some data need to be deleted to make room for future data as the flash starts filling up. HybridStore exploits a simple time-based data aging mechanism to discard the oldest data. When no space is available on the NAND flash to insert the current reading, HybridStore will locate the oldest segment and erase all the blocks in that segment. Then the minimum timestamp of all the readings currently stored in the system (i.e., `System.minT`) is updated. Since the NAND flash is organized as a circular array, wear leveling is trivially guaranteed. In addition, since segments are independent of each other, no garbage collection mechanism is needed here. On the contrary, other index schemes (e.g., [2,9,17]), require extensive page reads and writes to move valid pages within the reclaimed erase blocks to new locations, and maintain extra data structures in flash or RAM.

Note that we do not need to delete the pointers referencing the reclaimed segment from the skip list, even though they become invalid now. This problem is handled by the `select` algorithm (Line 14 and 32). Whenever a pointer with a timestamp smaller than `System.minT` is encountered, the `select` algorithm knows that this pointer is invalid and the query processing is completed successfully. On the contrary, for *each invalid index entry*, MicroHash [7] must load the referenced data page to learn that this page has been deleted and re-used, resulting in many unnecessary page reads.

4 Implementation and Evaluation

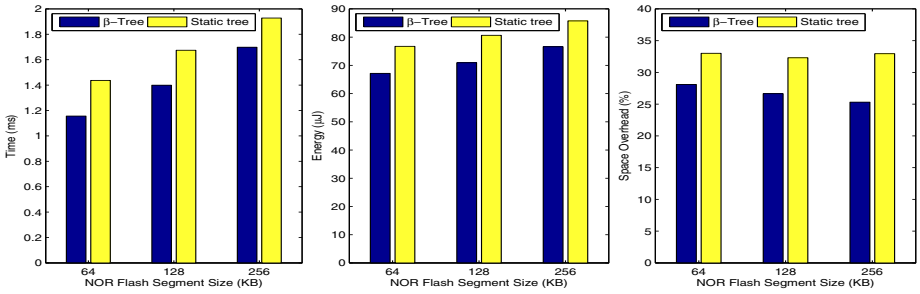
In this section, we describe the details of our experiments. HybridStore is implemented in TinyOS 2.1 and simulated in PowerTOSSIMz [15], an accurate power modeling extension to TOSSIM for MicaZ sensor platform. We additionally developed an emulator for a Toshiba TC58DVG02A1FT00 NAND flash (128MB), and a library that intercepts all communications between TinyOS and flash chips (both the NOR and the NAND flash) and calculate the latency and energy consumption based on Table 1. With all features included, our implementation requires approximately 16.5KB ROM and 3.2KB RAM, which is well below the limit of most constrained sensor platforms.

We adopt a trace-driven experimental methodology in which a real dataset is fed into the PowerTOSSIMz simulator. Specifically, we use the Washington Climate Dataset, which is a real dataset of atmospheric information collected by the Department of Atmospheric Sciences at the University of Washington. Our dataset contains 2,630,880 readings on a per-minute basis between 01/01/2000 and 12/31/2004. Each reading consists of temperature, barometric pressure, etc. We only index the temperature values and use the rest as part of the data records, each of which is of 32 bytes. To simulate data missing (e.g., reading drops due to the long latency during long queries and block erasures, or adaptive sensing), 5% readings are deleted randomly. For each kind of queries, 1000 instances are generated randomly and their average performance is presented here.

We compare HybridStore with MicroHash [7], Antelope [17] and the system in [2]. Since we do not have enough details to reproduce their complete experiments, we directly use the results reported in their papers if necessary. We use the same dataset as MicroHash, and fully implement the static bucket splitting scheme used in [7,17].

4.1 Insertions

We first insert all readings into the sensor mote and record the performance of each insertion. Fig. 2 shows the average performance of the β-Tree and the static bucket splitting scheme used in [7,17]. Compared to the β-Tree, the latter scheme consumes 13.24% more energy, induces 16.76% more space overhead, and results in 18.47% more latency on average. The latency and energy consumption of each insertion approximately equal to the write of 1.31 NAND pages and 0.91 NAND pages, respectively.



(a) Latency (b) Energy (c) Space Overhead

Fig. 2. Performance per insertion

Fig. 3 shows part of the timeline for insertions at the beginning when the NOR segment size is 64KB. Our key observations are as follows. First, although it is very energy-consuming ($26.8mJ$) to transfer the index from NOR flash to NAND flash, it only happens once every 3–4 days and is independent of the data record size. Second, during regular operation, each insertion consumes only $34.4\mu J$. When a reading is not within the current bucket range, the proper bucket can be located or created after traversing about 8–10 bucket headers in β-Tree, even when the current segment is almost full. Since there are about 236 buckets in the β-tree for each segment, our adaptive bucket splitting scheme generates a rather balanced tree. The points corresponding to around $0.15mJ$ and $1.2mJ$ additionally include the energy consumption to flush the write buffer to NAND flash and the Bloom filter buffer to NOR flash, respectively.

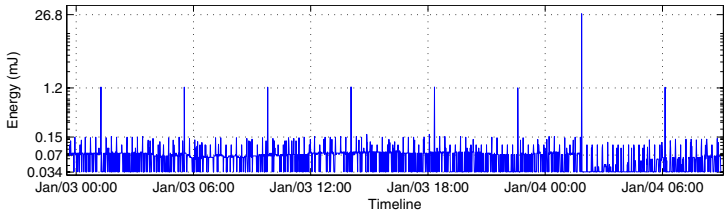


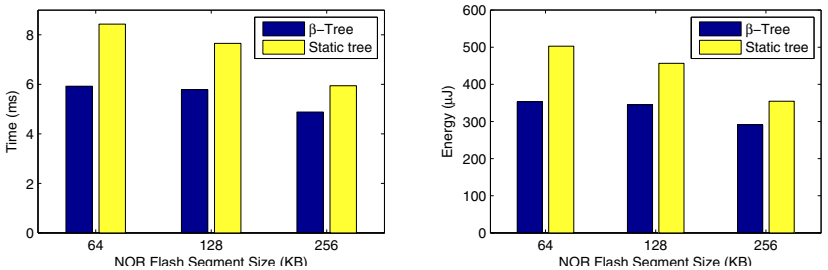
Fig. 3. Energy consumption of insertions

4.2 Time-Based Equality Queries

We also investigate the performance to process a time-equality query to find a record with a specific timestamp (i.e., $t_1 = t_2$). Fig. 4 shows that even though the query time window is quite large (i.e., over 2.5 million readings in 5 years), HybridStore is able to locate the record with about 5–6 page reads. Such a high performance can be achieved due to the following reasons. Firstly, the β-Tree improves the storage efficiency, resulting in fewer segments to store the same number of readings. Secondly, the skip list can locate the segment containing the required timestamp efficiently. Thirdly, the scale binary search can locate the page quickly because all readings are stored continuously in each segment, avoiding traversing through a block chain. Therefore, compared to a global index (e.g., MicroHash requires about 5.4 page reads to process such a query when the buffer size is 2.5KB, as shown in Fig. 14 in [7]), HybridStore has almost the same performance, while consuming less RAM. Compared to Antelope [17], HybridStore can achieve a much better performance if the same dataset is used.

4.3 Joint Queries: Time-Based Range and Value-based Equality

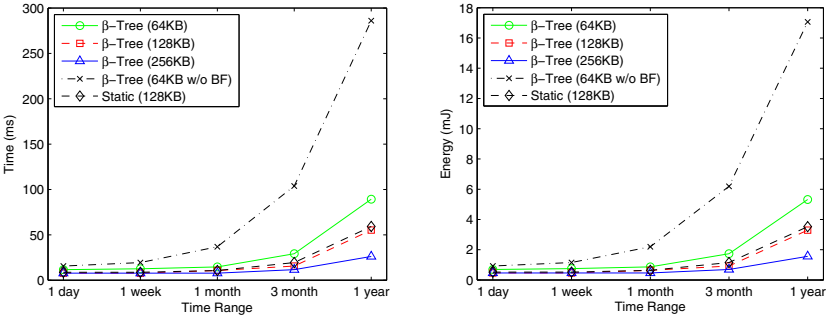
In this scenario, we study the impact of Bloom filter on joint time-based range and value-based equality queries. Fig. 5 shows the average performance per query to search nonexistent key values. We can observe that the in-segment Bloom filter can significantly improve the performance of value-based equality queries (more than 3 times improvement when the NOR segment size is 64KB and the time range is more than 3 months). In addition, the β-Tree can reduce the latency and energy consumption for queries involving large time window by about $4ms$ and $230\mu J$, respectively. Finally,



(a) Latency (b) Energy

Fig. 4. Performance of time-equality queries: HybridStore (β-Tree) v.s. Antelope [17]

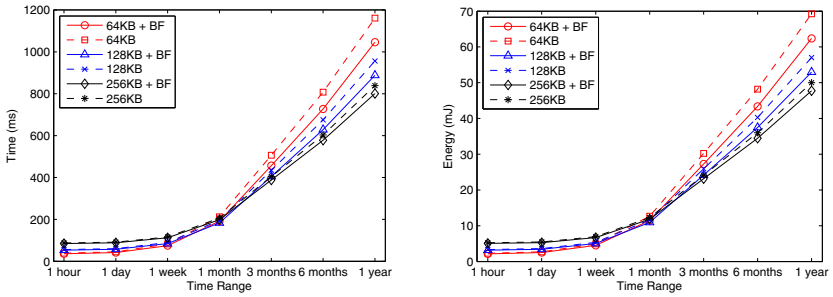
HybridStore is extremely efficient to check the existence of key values. When the NOR segment size is 256KB, HybridStore can decide the existence of a key value in over 0.5 million readings spanning one year time window in 26.18ms, consuming only 1.56mJ.



(a) Latency (b) Energy

Fig. 5. Impact of Bloom Filter on value-based equality queries for nonexistent keys

We also investigate the average performance per query to search existing key values, which is shown in Fig. 6. The key values vary in [40 °F, 60 °F]. We can observe that the in-segment Bloom filter can reduce the latency and energy consumption for queries involving large time window by 38–116ms and 3–7mJ, respectively. More importantly, HybridStore requires approximately only 826 page reads to get all readings with the given key value in one year time window when the NOR segment size is 256KB. Comparatively, MicroHash requires about 8700 page reads on average to search a given key value $\in$ [40 °F, 60 °F] in five years time window (inferred from Fig. 15 in [7]). Even if we assume that MicroHash can "intelligently" stop searching when a reading below the lower bound of the query time window is encountered, it still requires much more than 1740 page reads for one year time window, because many index pages and data pages are read unnecessarily.

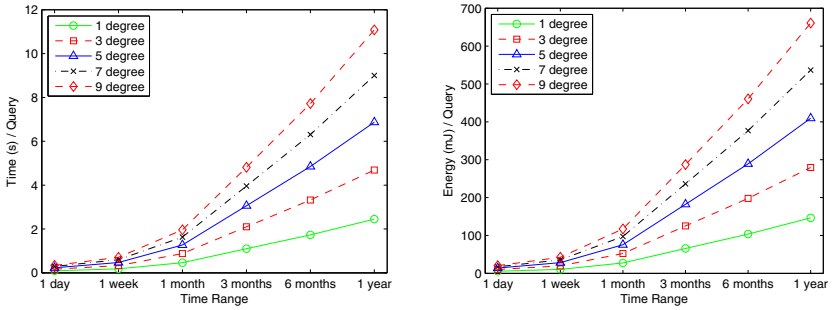


(a) Latency (b) Energy

Fig. 6. Impact of Bloom Filter on value-based equality queries for existing keys

4.4 Joint Queries: Both Time-Based and Value-based Ranges

In this scenario, we investigate the most common type of queries that involves both time windows and value ranges as selection predicates. Fig. 7 shows the average performance per query when the NOR segment size is 64KB. We can observe that HybridStore is extremely efficient to process such queries. When the value range is 1 °F and the time window is 1 month (typical queries, because readings in small time windows are more interesting), HybridStore can finish the query in 461.6ms, consumes only 27.5mJ and returns 2678 readings. For queries involving a large value range (e.g., 9 °F) and a long time window (e.g., 1 year), HybridStore can return 120,363 readings in 11.08s, consuming only 660.7mJ (92.04μs and 5.48μJ per reading on average). Compared to Antelope [17], since the NOR flash is much slower and less energy-efficient, Antelope will take about 20s to retrieve 50% readings from a table with only 50,000 tuples in a range query (shown in Fig. 8 in [17]).



(a) Total Latency per query (b) Total energy per query

Fig. 7. HybridStore performance per query of full queries

Another index scheme proposed in [2] can support range queries. It will consume about 40mJ on average to process a query with 5-degree range on about only 100,000

readings (about 13,000 readings are returned). Comparatively, HybridStore will consume 75.61mJ to return the same number of readings by processing the same query on more than 2.5 million readings. While our dataset size is 25 times larger than the dataset used in [2], HybridStore consumes only 89% more energy. Therefore, HybridStore is more energy efficient to support queries on large-scale datasets. Besides, the size of each reading in [2] is much smaller, which consists of only a timestamp and a temperature value (12B is enough, while our record size is 32B), resulting in much less data pages. Finally, their scheme requires much more RAM resource (close to 10KB, shown in Page 10 in [2]), because the per-partition B-tree index, interval table, and the *last page of each interval's list* must be maintained in RAM.

5 Conclusions

In this paper, we proposed HybridStore, an efficient data management system for low-end sensor platforms, which exploits both the on-board NOR flash and external NAND flash to store and query sensor data. Compared to existing works that can only support simple queries, HybridStore can process typical joint queries involving both time windows and key value ranges as selection predicates extremely efficiently, even on large-scale datasets. Our evaluation with a large-scale real-world dataset reveals that HybridStore can achieve remarkable performance at a small cost of constructing the index. Therefore, HybridStore provides a powerful new framework to realize *in situ* data storage in WSNs to improve both in-network processing and energy-efficiency.

Acknowledgements. We would like to thank our shepherd, Luca Mottola, and the anonymous reviewers for their insight and detailed feedback. This work is partially supported by DARPA and SRC through grant award 013641-001 of the FCRP, by the National Science Foundation (NSF) under grant award CNS-1035655, and by the National Institute of Standards and Technology (NIST) under grant award 70NANB11H148.

References

1. Agrawal, D., Ganesan, D., Sitaraman, R., Diao, Y., Singh, S.: Lazy-adaptive tree: an optimized index structure for flash devices. In: ACM VLDB, pp. 361–372 (2009)
2. Agrawal, D., Li, B., Cao, Z., Ganesan, D., Diao, Y., Shenoy, P.: Exploiting the interplay between memory and flash storage in embedded sensor devices. In: 16th IEEE Intl. Conf. on Embedded and Real-Time Computing Systems and Applications, pp. 227–236 (2010)
3. Atmel Inc.: AT45DB041B, http://www.atmel.com/Images/doc3443.pdf
4. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13(7), 422–426 (1970)
5. Kang, D., Jung, D., Kang, J.U., Kim, J.S.: μ-tree: an ordered index structure for NAND flash memory. In: ACM EMSOFT, pp. 144–153 (2007)
6. Li, H., Liang, D., Xie, L., Zhang, G., Ramamritham, K.: TL-Tree: flash-optimized storage for time-series sensing data on sensor platforms. In: ACM SAC, pp. 1565–1572 (2012)
7. Lin, S., Zeinalipour-Yazti, D., Kalogeraki, V., Gunopulos, D.: Efficient indexing data structures for flash-based sensor devices. ACM Trans. on Storage 2(4), 468–503 (2006)

8. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: TinyDB: an acquisitional query processing system for sensor networks. ACM TODS 30(1), 122–173 (2005)
9. Mathur, G., Desnoyers, P., Ganesan, D., Shenoy, P.: Capsule: an energy-optimized object storage system for memory-constrained sensor devices. In: ACM SenSys, pp. 195–208 (2006)
10. Mathur, G., Desnoyers, P., Ganesan, D., Shenoy, P.: Ultra-low power data storage for sensor networks. In: ACM/IEEE IPSN, pp. 374–381 (2006)
11. Mottola, L.: Programming storage-centric sensor networks with squirrel. In: ACM IPSN, pp. 1–12 (2010)
12. Nath, S.: Energy efficient sensor data logging with amnesic flash storage. In: ACM/IEEE IPSN, pp. 157–168 (2009)
13. Nath, S., Gibbons, P.B.: Online maintenance of very large random samples on flash storage. In: ACM VLDB, pp. 970–983 (2008)
14. Nath, S., Kansal, A.: FlashDB: dynamic self-tuning database for NAND flash. In: ACM/IEEE IPSN, pp. 410–419 (2007)
15. Perla, E., Catháin, A.O., Carbajo, R.S., Huggard, M., Mc Goldrick, C.: PowerTOSSIMz: realistic energy modelling for wireless sensor network environments. In: Proc. of the 3rd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks, pp. 35–42 (2008)
16. Pugh, W.: Skip lists: a probabilistic alternative to balanced trees. Communications of the ACM 33(6), 668–676 (1990)
17. Tsiftes, N., Dunkels, A.: A database in every sensor. In: ACM SenSys, pp. 316–332 (2011)
18. Yin, S., Pucheral, P., Meng, X.: A sequential indexing scheme for flash-based embedded systems. In: ACM EDBT, pp. 588–599 (2009)

Annotating Real-World Objects Using Semantic Entities

Henning Hasemann[1], Oliver Kleine[2], Alexander Kröller[1], Myriam Leggieri[3], and Dennis Pfisterer[2]

[1] Algorithms Group, TU Braunschweig, Germany
{hasemann,kroeller}@ibr.cs.tu-bs.de
[2] Institute of Telematics, Universität zu Lübeck, Germany
{kleine,pfisterer}@itm.uni-luebeck.de
[3] Digital Enterprise Research Institute, Galway, Ireland
myriam.leggieri@deri.org

Abstract. Due to protocols such as 6LoWPAN and CoAP, wireless sensor nodes have become an integral part of the Internet. While this makes their data available on the Internet, it still remains a strictly device-centric approach. However, one is usually interested in the phenomena observed by one or more devices and not in the sensors themselves. To bridge this gap, we introduce and evaluate the concept of Service-Level Semantic Entities (SLSE). They support the automatic semantic annotation of real-world objects based on networked sensor devices. SLSE continuously collect observations of sensors attached to some objects, and aggregates them to descriptions of the objects themselves. It then makes this augmented object data available as (semantic) linked data. This removes several layers of indirection in the semantic data, and allows to directly access properties of real-world objects.

Keywords: Internet of Things, Semantic Entity, Semantic Web, Linked Sensor Data.

1 Introduction

"The entity of interest is a cow", says Stephan Haller in the IoT Comic Book [18], highlighting a crucial aspect of the Internet of Things: When we are attaching networked sensors to things, we really are interested in the things themselves, not the sensors. Unlike computers, people immediately understand how observations made by the sensors are, to some extent, observations about these things.

For a long time, IoT research has tried to bridge the gap among real and digital world, with the aim to enhance each other. The Constrained Application Protocol (CoAP) [19] allows accessing services on resource-constrained machines such as sensor nodes from the Internet, providing standardized and easy-to-use methods to make sensor data available. This data, when represented as semantic information in RDF[7], becomes part of the Semantic Web.

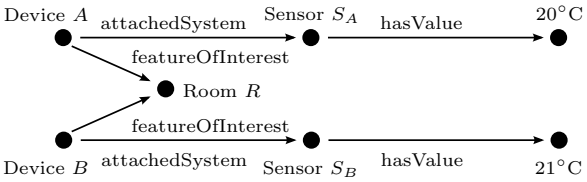
Unfortunately, the result is a strictly *device-centric* view. For an example, consider two devices A and B, monitoring temperature in a room R. One can

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 67–82, 2013.

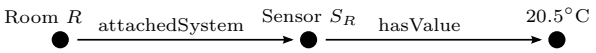
connect to these devices' services, receive their description in RDF. The resulting information is shown in Figure 1(a).

Usually a user or an application will be interested in the *room* itself, the devices are merely infrastructure and should be invisible. In this paper, we present *Service-Level Semantic Entities* (SLSE), a system to remedy this discrepancy. It was developed within the SPITFIRE project [16], and aims at providing semantic descriptions for real-world objects using RESTful web services.

In the above example, SLSE will automatically create a service endpoint (URI) for the room itself. Querying this service returns a RDF description of the room, as shown in Figure 1(b). Effectively this turns the room into what we call a *Semantic Entity*: A real-world object, represented by a URI, that can offer services, and that describes itself using RDF. This eradicates the separation between real and digital world: A room, or a cow, or any other object, participates in the Semantic Web, compatible with the growing body of semantic world knowledge available as Linked Open Data (LOD) [14], enabled by supporting devices that stay in the background.



(a) Original device-centric description.



(b) Improved representation using an SE.

(c) Ideal outcome after further processing.

Fig. 1. Using SLSE to create Semantic Entities

The result of this process is a semantic description of an observed object with one virtual sensor for each observed property that might aggregate a larger number of physical sensors on different sensor nodes (Fig. 1(b)). Depending on the further use of the semantic descriptions it might be worthwile for a client to do an additional inference step to get rid of the virtual sensors and semantically describe their observations as properties of the observed object (Fig. 1(c)).

The rest of this paper is organized as follows: Section 2 summarizes related work. In Section 3, we provide details on the architecture of SLSE and describe

optimizations that let the system answer queries efficiently. This is followed by a performance evaluation, using stress tests and a real-world example. Finally, Section 5 contains more examples to show how SLSE actually can improve applications.

2 Related Work

To make smart objects being Internet-connected with each other, a unified network layer is required. Using CoAP/HTTP, such connectivity has already been enabled. There are important open challenges, i.e.:

1. Resource constraints and hardware limitations need to be considered in the application layer
2. Devices should be able to understand the data that is exchanged by the unified network layer yielding semantic interoperability and reasoning
3. Devices should be discoverable, taking into account both capabilities and constraints of interest

The Open Geospatial Consortium's (OGC) Sensor Web Enablement (SWE) [15] project has defined a framework to publish and access sensor data using XML-based protocols and application programming interfaces (APIs), SOS [9]. The use of XML means that ad-hoc mapping of different schemas are needed to integrate data. Here, neither semantic interoperability is enabled nor is reasoning supported. As an attempt to account for that issue, a semantic extension to SOS has been developed, SemSOS [23].

Semantic Web technologies address such shortcomings by using machine-understandable descriptions of resources, which do not require any ad-hoc schema. The de-facto standard as a representation model is RDF and the meaning of each term can be gathered automatically by checking the corresponding vocabulary definition. The Semantic Sensor Web [20] studies the application of these technologies to enable the Sensor Web. The OGC defined a vocabulary for sensors, SensorML [5], which has been included into the SSN-XG ontology [22] realized by the W3C Semantic Sensor Network Incubator Group. That ontology aims at providing cross-domain concepts for sensors, inspired by several domain-specific ontologies that existed before. It allows annotating sensor-related features such as deployment, observations and measurement capabilities, thus enabling the automation of further tasks like fine-grained discovery and maintenance scheduling. One example of those tasks is the search for sensors which are observing wind direction with a specific accuracy level.

A further evolution of simple semantic annotations consists in following the Linked Data principles. These stress on identifying resources by HTTP-dereferenceable URIs, which provide useful information and contain meaningful semantically annotated links to other resources. A set of this kind of data is already available, publicly accessible and known as the Linked Open Data (LOD) cloud. Automated integration and reasoning can be performed on top of it. Its size has been increasing exponentially over the last few years.

Several research projects have followed the Semantic Sensor Web vision and are currently using the W3C SSN-XG ontology. Examples include SENSEI [17], the Spanish Meteorological Agency [21], the Sensor Masher [12], the work done by the Kno.e.sis Center [3], CSIRO [1] and in the 52 North projects [10]; SemSorGrid4Env [6] and Exalted [2]. Their level of abstraction is limited to the one provided by the ontological concepts used; while there is no human-intuitive abstractions for things and their correspondence with real world ones and sensors. We address these limits by proposing Semantic Entities as a further abstraction for things, to provide people with a more intuitive approach to handle raw data.

3 System Architecture

Consider a set of sensor devices observing some real-world object such as a cow. Our system, SLSE, provides a direct interface to this object by offering enriched semantic services for it. The resulting object is called *Semantic Entity* (SE), the sensors defining it are called *Base Entities* (BE). The system performs three steps:

1. Create a URI and a RESTful service endpoint for the SE. In the optimal case, both have the same HTTP URL so that one can directly connect to the RESTful service using the SEs URI. In many cases, the SEs URI is located in some different domain than the ones issuable by the SLSE system such as a URI from DBpedia. Then a placeholder service will be created, and the two addresses are made semantically equivalent using a `<http://www.w3.org/2002/07/owl#sameAs>` statement.
2. Process sensor data from sensors that monitor the SE and rewrite them to refer directly to the SE. In doing so, the SE has a semantic description and there is no need to directly use sensors and possibly sensor devices.
3. Act as a service proxy for the SE. Incoming service invocations for the created endpoint will either be answered by the SLSE in lieu of the SE or are relayed to the sensors observing the SE.

The SLSE embeds into the larger SPITFIRE system as by interfacing with the *Smart Service Proxy* (SSP) for both querying sensor node descriptions and providing service endpoints to the constructed SEs.

The SSP is comprised of a frontend that defines its RESTful interface towards the Internet and one or multiple backends facing the actual sensors. The backends are exchangeable, allowing for arbitrary sensor types and sensor-like data sources. A backend implementation collects data measured by sensors and adapts the communication protocols as required by the sensors. The collected data are provided via the SSPs RESTful API allowing access to sensor descriptions using a URI, which resolves to an RDF document describing the sensors, their measurements, and additional information such as location, type, etc. For a discussion of a preliminary version of the SSP, we refer the reader to Bimschas et al. [4].

The SLSE itself acts as a SSP backend describing Semantic Entities (instead of individual sensor nodes) to the querying client. In order to collect the node

descriptions that are to be aggregated into a Semnatic Entity, the SLSE in turn connects to a number of SSP instances or other data sources in the role of a querying client. See Figure 2 for an illustration.

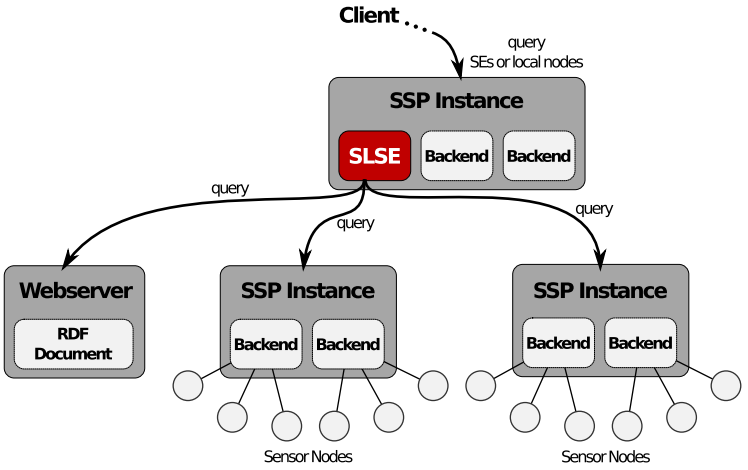


Fig. 2. Interaction of SLSE and other SPITFIRE components

In our implementation of the SLSE backend, data sources called *Base Entities* (BE) are polled regularly. These data sources can be SSP instances, or anything else that provides a compatible RDF description such as RDF documents on a Webserver. Based on a dynamic set of rules, the SLSE backend then maintains a list of SEs. In addition to the representation of their corresponding real-world object, SEs are annotated using the SPITFIRE ontology[1]. This ontology is based on the alignment of Dolce Ultralite[2] (DUL), the W3C SSN-XG ontology and the Event Model F ontology [3]. It has been defined to allow descriptions for cross-domain sensor-related data and context [13]. The alignment with an upper level ontology like DUL and the SSN-XG results in an easier integration with a broader range of different annotations.

3.1 Data Representation

Several formats have established for describing sensors including their capabilities and measurements semantically. Most of these formats however provide a restricted set of primitives for describing facts, usually bound to a specific domain of application. Also these formats are specific to the area of sensor networks and

[1] http://spitfire-project.eu/cc/spitfireCC_n3.owl
[2] http://ontologydesignpatterns.org/wiki/Ontology:DOLCE%2BDnS_Ultralite
[3] http://www.uni-koblenz-landau.de/koblenz/fb4/AGStaab/Research/ontologies/events

thus do not allow direct integration of the obtained descriptions with existing data in the World Wide Web.

In this work we investigate the use of the *Resource Description Framework* (RDF) for describing the properties of sensor nodes and ultimately Semantic Entities. This approach offers a number of interesting properties:

- As RDF is also the data format of the Linked Data cloud [14], it enables linking descriptions of sensors and Semantic Entities with facts from arbitrary fields of interest available in the Semantic Web.
- Due to its universality in terms of described content, RDF enables sensor node descriptions to contain arbitrary application specific information. I.e. a node monitoring a device can hold a semantic description of that device, its surroundings and/or knowledge about its high-level status such as "out of order" and incorporate that information into its workings.
- Finally, a language like RDF which allows arbitrary facts to be expressed enables describing sensor nodes as well as generated Semantic Entities which – as they are a more abstract concept – could not be fully described using the terminology of sensor nodes.

3.2 Sensor Network

The architecture of the Smart Service Proxy allows a multitude of sensor network implementations. The only requirement being that the backend implementation is able to

1. Inform the system about joining and leaving nodes
2. Provide semantic sensor node descriptions upon request

As long as these properties are ensured, any choice of routing- or data collection mechanism is possible. This leaves the implementer free to choose a mechanism that is tailored for the demands of the network infrastructure in use.

One example of such a network is the architecture currently being developed in SPITFIRE: Nodes hold compressed RDF descriptions of themselves which are accessible via the Wiselib RDF Provider [8] over a number of communication protocols such as CoAP over 6LoWPAN [11]. Using a discovery mechanism the respective backend implementation gets informed about the state of participating nodes and queries their description on demand, caching as appropriate depending on the `Max-Age` HTTP headers of incoming requests. Another implementation (which we will use for evaluation in Section 4) queries node descriptions which are available in a proprietary format from an existing data server. In a second step, it then converts these descriptions to RDF using a set of predefined rules and passes them on to the SSP Core.

Usually a SSP backend will involve some kind of caching of node descriptions which also provides a certain grace period for unresponsive devices. If a device is unreachable for long enough however a backend should report its leaving to the SSP Core so it will not be reported queryable anymore. As the details of this process are highly specific to the sensor network implementation, backends might impose different behaviour in this context.

3.3 SSP Core

The SSP consists of a number of handlers arranged in a stack, each responsible for a specific task to efficiently respond service requests (cf. Figure 3). The SSPs core handlers are those who either respond to incoming requests using cached data or – in case of data that is expired or not cached – forward requests to the appropriate backend. In this stack of handlers, only the necessary ones, in both directions (upstream and downstream) are invoked. There are handlers involved for upstream and downstream such as `Statement Cache` and `Formatter` and others only for upstream such as `Entity Manager` and `Backend`.

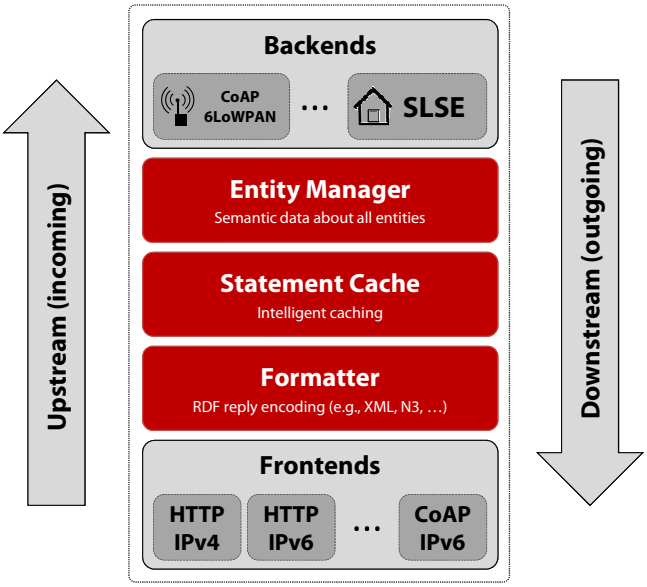


Fig. 3. Architecture of the Smart Service Proxy

The following list describes these handlers in more detail:

- The *Formatter* is the first upstream handler to deal with incoming requests. Its task for the upstream direction is to record the acceptable MIME types as specified in the HTTP request. Downstream, it converts data – which is the payload of HTTP responses – to one of these MIME types.
- The *Statement Cache* maintains an internal cache. If it finds an unexpired state of a requested resource, it sends this state to the Formatter to answer requests immediately without involving any other handler. Otherwise, it sends the request further upstream.
- The *Entity Manager* is responsible for administering backends. Backends are upstream handlers that are registered with the Entity Manager and provide a backend-specific list of resources, which are updated continuously. Thus

the Entity Manager is aware of all resources, i.e., URIs of all resources of all registered backends, and is thus able to select the appropriate backend to put on top of the handler stack at runtime.
– As described in Section 3.4 for the case of the SLSE backend, the backend is responsible to provide actual resource states. Such resources could be either SEs or raw sensors. Once the current status is determined, the backend sends the status including an expiry date downstream, making it pass through Statement Cache and Formatter.

3.4 SLSE Backend

The SLSE backend accesses and aggregates semantic descriptions of individual sensors, creating the semantic data for SEs. The architecture of the backend is shown in Figure 4 and comprises three major components:

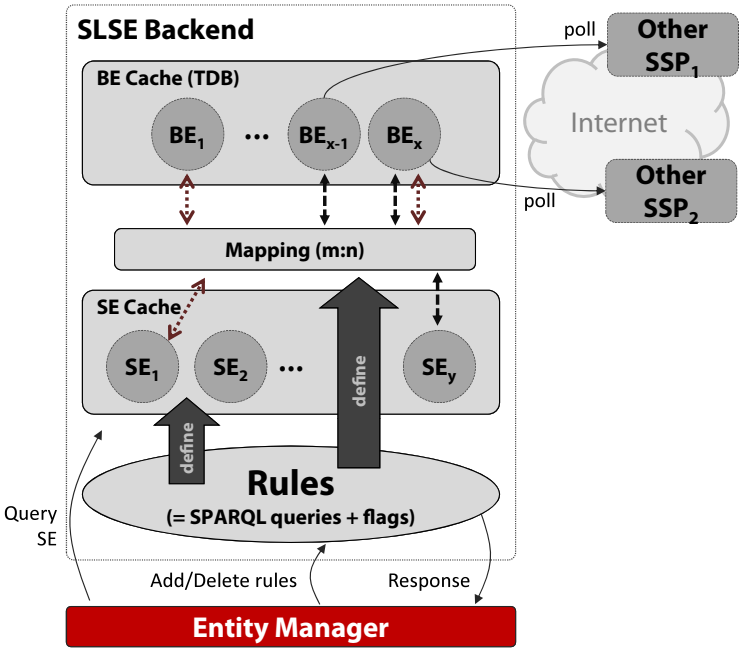


Fig. 4. Structure of the SLSE backend

– The *BE Cache* holds descriptions of all known BEs. It is implemented as an Apache Jena TDB [4] dataset. Each BE is stored as a named model, using its URI. This enables later retrieval of individual BEs and efficient queries over the union of all BEs.

[4] http://incubator.apache.org/jena/documentation/tdb/

- The *Rules Pool* holds a set of SE-defining rules, each of which is defined by a SPARQL query and configuration flags. The rules define how BEs are aggregated to SEs and what changes in BE descriptions cause SE updates. This is done by executing each rules SPARQL query on the BE Cache and interpreting the response as a BE↔SE mapping. The details on the update behavior depend on configuration flags (see below). The BE↔SE-mapping is stored bidirectionally, i.e., SEs keep track of their BE members, and vice versa.
- The *SE Cache* holds all constructed SEs with sensor values and cached models. This cache enables the system to only recreate a SE (by re-evaluating the SPARQL query) when BE data changes in a way that invalidates the cached SE with a certain probability.

Backend Operation

Polling BEs. A dedicated polling thread checks all known data sources periodically. Usually SSPs provide access to the sensors themselves. However, any HTTP resource following the desired RDF format can be used here. First it queries `/.well-known/servers` to receive a list of known entities. This is compared to the last known entity-list for that source and yields a list of added, deleted or changed BEs. Afterwards, each BE description is updated.

Rule Creation. To define one or many SEs, a rule set is inserted into the system by invoking a RESTful web service. It consists of a SPARQL query and configuration flags. The SPARQL query selects the variables `?element` and `?entity`. The result of this query defines which BEs (`?element`) are mapped to which SLSEs (`?entity`). For an example query, see Figure 5. The configuration flags contain hints on the structure of the created SEs, enabling system-internal optimizations. Currently, we support the flags `not-value-dep`, indicating that the SEs structure does not depend on the numerical values of sensor readings and `decomposable`, indicating that the query can be decomposed for individual BEs. In Section 3.4, we describe the effects of these flags in more detail.

After such a request is received, the system will run the selection query on all known BEs and add the results to the current mapping. Every time the description of a BE changes, or a new BE is discovered during polling, the mapping is invalidated. After the polling cycle is complete, the query is executed on the union of all BEs and the mapping is updated with the result.

```
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>

SELECT ?element ?entity WHERE {
    ?element ssn:attachedSystem ?sensor .
    ?sensor ssn:featureOfInterest ?entity .
}
```

Fig. 5. Rule for constructing SLSEs based on feature of interest

SE Creation and Update Management. A rule is executed by applying its selection query to the union model of all known BEs. The result is a set of pairs of BE URIs (`?element`) and the URI of the object described by the resulting SE (`?entity`). The system generates a SE for each new value of `?entity`, assigns a URI to it, and maps it to `?element`. If `?entity` points to an existing SE, its mapping is updated accordingly.

SEs keep track of all sensor values of their BEs and an aggregated value for each observed property. Currently this aggregation defaults to calculating the average; this can be changed easily if needed as cached data of all sensors is available to the SLSE backend at all times. Whenever a BE is added or removed to a SE, the aggregated value for all affected observed properties is updated efficiently.

Configuration and Optimization

The SLSE backend contains several optimizations that exploit knowledge on the SEs structures to increase the system's efficiency. In the following, we describe the currently implemented ones.

Decomposable Rules. In the majority of usable rule sets, SPARQL queries contain no dependencies between BEs. If this is the case, it is possible to execute the query on each BE individually. An example for a non-decomposable rule is "*every sensor sharing property x with another sensor*". If a user marks a rule as `decomposable`, the SLSE backend executes the query directly on individual BE models when they are polled. This can improve performance, as the queried model is much smaller.

Rules Independent of Sensor Values. Usually, SEs are created by properties such as location or features of interest. Rarely, a rule is based on actual sensor values. As sensor values change more frequently than the rest of a sensors description, this observation can be used for optimizations. If the corresponding flag is set, the performance of SE updates is improved by re-executing the rule query only when information other than sensor values has changed in a BE update. This is especially valuable in combination with decomposable rules, as it allows for individual BEs to decide whether they need to be re-queried thereby increasing performance when only a few descriptions change in a poll cycle.

Parallelization. In normal operation mode, the BE cache is updated sequentially with locking of the full TDB. However, when there are many different BE proxies, it is desirable to poll them in parallel. This leads to shorter poll times, at the expense of slowing down requests during those polls as the polling threads compete for database access.

4 Evaluation

In this section we will demonstrate the system in a real-world example that involves internal and external data sources, SPARQL queries and actual sensor data, playing together to semantically answer a complex semantic question.

4.1 Experiment Scenario and Setup

To evaluate system behavior in a real-world application, we use the following fictional scenario: ACME offers maintenance services for cold storage rooms in restaurants. The cold rooms are equipped with temperature sensors. Assume one of ACME's maintenance teams is currently idle, being somewhere in the German state Lower Saxony. Then a dispatcher needs an answer to this query (assuming a bad temperature indicates a fault):

> "Find all the cold rooms in Lower Saxony, where the temperature is above −18° Celsius, where ACME has a support contract with the owner."

Such a query can be implemented using the SLSE, using the following data sources (see Figure 6):

- Semantic Entities: For each cold room, we create one SE, which is linked to the restaurant and its location. This dataset is remotely accessed and open.
- Dataset of contracts and clients of ACME. This dataset [5] is locally accessed and closed for privacy reasons.
- Dataset of the relations between locations, say, using Geonames. This dataset is remotely accessed and open.

To evaluate this scenario, we use sensor data from TU Braunschweig. We have access to four rooms that are monitored by 300+ sensors of different kinds, and use these to stand in for cold rooms. They have a temperature of around +20°C, so we adapt the query.

Now ACME can execute the above search using a single SPARQL query involving an SLSE for the rooms, a local file for the contracts, and an external database for the locations. The query is shown in Figure 7.

Neither the rooms nor GeoNames datasets provide a SPARQL endpoint or a named graph, so we have to import each resource explicitly using FROM. They are remotely accessed, so the query is realistic for the scenario.

[5] ACME contracts and clients - http://spitfire-project.eu/example/acme_internal.rdf#ACMECorp

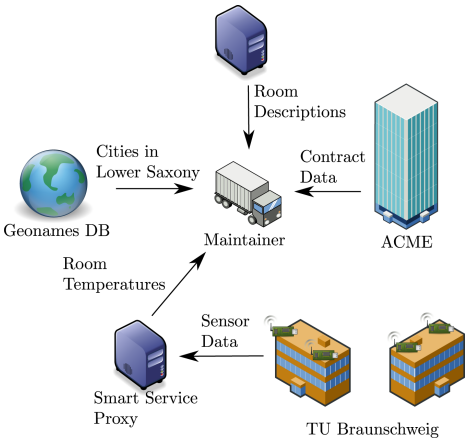


Fig. 6. Illustration of the query scenario

```
PREFIX  spitfire: <http://spitfire-project.eu/cc/spitfireCC_n3.owl#>
PREFIX  ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX  mads: <http://www.loc.gov/mads/rdf/v1#>
PREFIX  gn: <http://www.geonames.org/ontology#>
...

SELECT  ?sem_entity_room
FROM <http://spitfire.ibr.cs.tu-bs.de/be-0002/rooms-bs.sytes.net
/static/descriptions/Room1#>
...
FROM <http://spitfire.ibr.cs.tu-bs.de/be-0002/rooms-bs.sytes.net
/static/descriptions/Room4#>
FROM <http://rooms-bs.sytes.net/static/descriptions>
FROM <http://spitfire-project.eu/example/acme_internal.rdf>
FROM <http://sws.geonames.org/2862926/about.rdf>
FROM <http://www.geonames.org/2945024/about.rdf>

WHERE {
  ?sem_entity_room ssn:attachedSystem ?sensor .
  ?sensor dul:hasValue ?val .
  ?sem_entity_room owl:sameAs ?room .
  ?room rdf:type spitfire:ColdRoom .
  ?room dul:isPartOf ?org .
  ?org dul:hasLocation ?location .
  ?sensor ssn:observedProperty <http://dbpedia.org/resource/Temperature> .
  ?location gn:parentADM1 <http://sws.geonames.org/2862926/> .
  <http://spitfire-project.eu/example/acme_internal.rdf#ACMECorp>
     mads:hasAffiliation ?org .

  FILTER ( ?val > "-18" )
}
```

Fig. 7. SPARQL Query for the cold rooms application

The query selects semantic entities

1. In which something is sensing Temperature with values higher than -18 Celsius degree
2. Whose correspondent real world room is a cold one and is part of a building
3. Such building is located somewhere in Lower Saxony

4.2 Experiment Results

To test this use case, we ran the SPARQL query using a publicly available SPARQL engine[6]. This is a live demo of the OpenLink Virtuoso server, with a typical response time of around $.5 - 1$ seconds. In an actual company, a high-performance local server would be used. Query execution was triggered every 100ms. The CDF of the response times is shown in Figure 8.

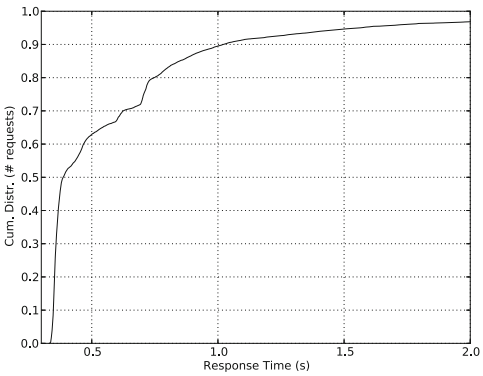


Fig. 8. CDF for response times to the cold rooms application query

It is clearly visible that the system achieves sufficient speed, the response time is typically (at the 90th percentile) below a second, i.e., it is as fast as one could wish for using an open SPARQL engine. Obviously fusing SEs through the SLSE with additional—both public and private—data sources is a viable approach to enable this application.

5 Applications

The application scenario used in Section 4.1 is only one example of the use-cases in which our approach is beneficial. Now, we present a selection of interesting application scenarios. These are based on a survey [18] by the Internet of Things Initiative[7].

[6] http://demo.openlinksw.com/sparql/
[7] http://www.iot-i.eu/

Smart City: Waste Management In this scenario, trash bins are equipped with sensor nodes and RFID readers. Waste bins authorize people, detect waste thrown into them, and monitor their fill level. This data is used to determine optimal waste collection routes, to perform billing, and to reward *green households* that produce only little waste. In addition, data about ambient conditions could be collected such as temperature, air quality and noise to detect hot-spots in a city. All this information is gathered by different sensors, we create one Semantic Entity for every trash bin. This abstraction eases data merging and reasoning.

As a result, we can alter the number of sensors per trash bin later for accuracy or reliability reasons without changing a single line of application code. During application development we gain an attractive level of abstraction: We can address the trash cans directly and do not need to even be concerned with the fact that their state is measured using sensor nodes.

Healthcare. In ageing societies, where the money available for elderly care decrease steadily, it becomes essential to support people to live as independently as possible. IoT technology can help by tracking locations and data about the personal well-being of people. One way to achieve that is monitoring body functions such as the heart rate. In a case of emergency, rescue personnel can be alarmed automatically and relevant medical data is linked immediately to this person. Healthcare operators should not have to check all the sensors on the elderly persons body manually.

A person in such a scenario might be tracked by sensors of different kind: Cameras, IR motion sensors, pressure sensors in the floor to only name a few. The vendor of such a healthcare system however can not know at programming time how exactly the involved sensor nodes might be installed. However by having them output semantic descriptions in sufficient detail, the installer of the system can use SLSE together with self-written, maybe even private semantic data sources like RDF documents in order to extract the person or other interesting objects as semantic entities. A physician or healthcare associate can then access these entities and maintain applications specifically tailored for the condition of the patient.

Smart Energy and Home Automation. To reduce CO_2 emissions, a cleaner generation of power and a more sensible consumption of energy is required. On the consumer side, consumption should be more balanced to reduce the peak amount of energy required such that fewer power plants are needed. To achieve this, devices such as air conditions or washing machines could negotiate their active cycles on a city or nation-wide scale such that the lowest consumption at the lowest price is achieved. In addition, rooms could detect the presence of people and automatically switch off consumers such as light and TV if nobody is present or adapt the required heating, etc. In this scenario data about many different subjects would need to be merged. In particular data from sensors placed in outdoor/indoor sites and attached to energy consuming devices, would merge with data about the energy provider companies such as energy bill prices. For instance to get data about everything related to a heater and merge it with

outdoor temperature and energy bill prices, it would be enough to just query the semantic entities for heaters, outdoor places such as the street of a city and energy provider companies. Therefore we see constructing SEs for every household device as a first step to achieve this vision.

6 Conclusion and Future Work

We have presented *Service-Level Semantic Entities* (SLSE), a system to aggregate semantic data from sensors to information about real-world things, and to provide RESTful services to query them directly. In our experiments we have showcased and evaluated a real-world application example, showing the usefulness and functionality of the system.

SLSE is useful in its current state; however there are two important improvements that we will address next:

1. Currently, SLSE can aggregate spatially (fuse sensor streams), but not temporally. A future SLSE should be able to deal with short-lived sensor-object contacts, for example when goods in a manufacturing line pass sensors, to create an SE for every produced item.
2. The system does not use the processing capabilities of the sensor devices, but rather constructs SEs on a central server. Instead, it should be possible that the devices themselves form the SE in a completely autonomous and distributed way. This might also reduce the currently necessary traffic for data collection.
3. Some of the exploited properties of semantic descriptions and SE construction rules such as decomposable rules currently need to be annotated manually. A possible improvement would be automatic recognition of those properties and activation of appropriate optimizations.

In the IoT Comic Book, Stephan Haller is also quoted with "[...] most people would agree that we do not connect directly to the cow!". We disagree.

References

1. Commonwealth scientific and industrial research organisation, http://www.csiro.au/science/Sensors-and-network-technologies.html
2. EXALTED Project (EXpAnding LTE for Devices), http://www.ict-exalted.eu
3. Ohio center of excellence in knowledge-enabled computing, http://knoesis.wright.edu
4. Bimschas, D., Hellbrück, H., Mietz, R., Pfisterer, D., Römer, K., Teubler, T.: Middleware for smart gateways connecting sensornets to the internet. In: Proceedings of the 5th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks, MidSens 2010, pp. 8–14. ACM, New York (2010), http://doi.acm.org/10.1145/1890784.1890787
5. Botts, M., Robin, A.: OpenGIS Sensor Model Language (SensorML) Implementation Specification. Tech. rep., OGC (July 2007)

6. Gray, A.J.G., Galpin, I., Fernandes, A.A.A., Paton, N.W., Page, K., Sadler, J., Kyzirakos, K., Koubarakis, M., Calbimonte, J.P., Castro, R.G., Corcho, O., Gabaldon, J.E., Aparicio, J.: Semsorgrid4env architecture. Technical report, SemsorGrid4Env project consortium (December 2010), http://eprints.soton.ac.uk/272700/
7. Group, W.R.W.: Resource Description Framework (RDF), Online version at http://www.w3.org/RDF/ (February 10, 2004)
8. Hasemann, H., Kröller, A., Pagel, M.: RDF Provisioning for the Internet of Things. In: Proceedings of Internet of Things 2012 International Conference (IoT 2012) (2012)
9. Henson, C.A., Pschorr, J.K., Sheth, A.P., Thirunarayan, K.: SemSOS: Semantic sensor Observation Service. In: International Symposium on Collaborative Technologies and Systems, pp. 44–53 (2009)
10. Jan Kraak, M., Adam Sliwinski, A., Wytzisk, A.: What happens at 52N? In: An Open Source Approach to Education and Research (2005), http://52north.org/
11. Kushalnagar, N., Montenegro, G., Schumacher, C.: IPv6 over low-power wireless personal area networks (6LoWPANs): Overview, assumptions, problem statement, and goals. Tech. rep., IETF Secretariat (2007), http://www.rfc-editor.org/rfc/rfc4919.txt
12. Le-Phuoc, D., Hauswirth, M.: Linked open data in sensor data mashups. In: 2nd International Workshop on Semantic Sensor Networks (SSN 2009) (2009)
13. Leggieri, M., Passant, A., Hauswirth, M.: A contextualised cognitive perspective for linked sensor data. In: 3rd Intl. Workshop on Semantic Sensor Network (2010)
14. Linking Open Data Community: The linking open data cloud diagram, Online version at http://richard.cyganiak.de/2007/10/lod/ (September 19, 2011)
15. OGC - Open Geospatial Consortium: Sensor Web Enablement (SWE) (2010)
16. Pfisterer, D., Römer, K., Bimschas, D., Kleine, O., Mietz, R., Truong, C., Hasemann, H., Kröller, A., Pagel, M., Hauswirth, M., Karnstedt, M., Leggieri, M., Passant, A., Richardson, R.: SPITFIRE: toward a semantic web of things. IEEE Communications Magazine 49(11), 40–48 (2011)
17. Presser, M., Barnaghi, P., Eurich, M., Villalonga, C.: The sensei project: integrating the physical world with the digital world of the network of the future. IEEE Communications Magazine 47(4), 1–4 (2009)
18. Presser, M. (ed.): Inspiring the Internet of Things! Alexandra Institute (2011), http://www.alexandra.dk/uk/services/Publications/Documents/IoT_Comic_Book.pdf
19. Shelby, Z., Hartke, K., Bormann, C., Frank, B.: Constrained application protocol (CoAP) (CoRE working group), Online version at http://www.ietf.org/id/draft-ietf-core-coap-08.txt (November 01, 2011)
20. Sheth, A., Henson, C., Sahoo, S.: Semantic Sensor Web. IEEE Internet Computing, 78–83 (2008)
21. Villazn-Terrazas, B., Vilches-Blzquez, L.M., Corcho, O., Gmez-Prez, A.: Methodological guidelines for publishing government linked data. In: Wood, D. (ed.) Linking Government Data, pp. 27–49. Springer, New York (2011)
22. W3C Semantic Sensor Network members: W3C Semantic Sensor Network Incubator Group (2010), http://www.w3.org/2005/Incubator/ssn/
23. Wei, W., Barnaghi, P.: Semantic Annotation and Reasoning for Sensor Data. In: Barnaghi, P., Moessner, K., Presser, M., Meissner, S. (eds.) EuroSSC 2009. LNCS, vol. 5741, pp. 66–76. Springer, Heidelberg (2009)

Data-Aware, Resource-Aware, Lossless Compression for Sensor Networks

Rachel Cardell-Oliver[1], Stefan Böttcher[2], and Christof Hübner[3]

[1] The University of Western Australia, Perth, Australia
rachel.cardell-oliver@uwa.edu.au
[2] Universität Paderborn, Germany
stb@uni-paderborn.de
[3] University of Applied Sciences Mannheim, Germany
c.huebner@hs-mannheim.de

Abstract. Compressing sensor data benefits sensor network applications because compression saves both transmission energy and storage space. This paper presents a novel lossless compression algorithm for sensor networks that is both data-aware and resource-aware. The DARA algorithm provides high compression ratios and also has a small memory footprint and efficient execution well within the range of sensor nodes. It is demonstrated that data-awareness, that is exploiting the structure of sensor data, is an important contributor to compression performance. The practicality of the DARA algorithm is demonstrated by an application in which sensor nodes use a phone modem to transmit a daily digest of nine sensor data streams in a single SMS message.

1 Introduction

Compressing sensor data benefits sensor network applications because compression saves transmission energy and storage space. Energy is saved when the quantity of data transmitted by network nodes is reduced. Compression also reduces on and off-node storage requirements so contributing to the goal of building long-life, unattended sensor networks. Any sensor network application that can tolerate delays in the delivery of their data can benefit from on-node compression. Such applications include environmental networks, particularly in remote areas; networks using opportunistic data collection by mobile data mules; bio-data sensed by body implant sensors; networks that transmit code for node reconfiguration; and applications that require efficient off-network storage of their data. Even for delay-intolerant applications, compressing current data can save transmission energy and storage since shorter messages are generated.

Compression algorithms for text and images have been widely studied, but only a few have been proposed that are suitable for the constraints of sensor networks. We argue that successful compression of sensor data streams requires both *data-awareness* of the structural properties of the raw data and *resource-awareness* of the constraints of the sensor nodes on which compression algorithms will be run.

This paper presents the design and evaluation of a generic algorithm, DARA, for on-node, lossless compression of sensor readings. The main contributions are as follows. (1) DARA is data-aware: it maps raw sensor data streams (that is, timed series

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 83–98, 2013.

of readings from a particular sensor) to an equivalent form that maximises compression. (2) DARA is resource-aware: it uses Huffman coding with small codebooks, and allows for previously unseen or erroneous readings, as well as lost transmissions. (3) DARA is flexible and practical: analysis of different types of sensor data streams and evaluation of an implementation show that it offers better compression ratios than existing algorithms while still having low space and time complexity, and being adaptive to real-world conditions.

The paper is organised as follows. Section 2 reviews related work on compressing sensor data streams and gives an overview of our approach. Section 3 investigates data-awareness by testing the compressibility of different data models. Section 4 discusses the resource-aware features of the DARA algorithm. To illustrate the practicality of our algorithm, Section 5 presents the implementation of a sensor network application that reads heterogeneous sensor data streams and delivers compressed data once per day in an SMS message. This application requires a high degree of compression, and so is a good demonstration for the performance of the DARA algorithm.

2 Background and Related Work

Compression algorithms transform a stream of *input symbols* into a stream of *encoded symbols* by mapping one or more symbols in the input onto a code word [2]. The compression ratio *CR* for a particular message m is defined by

$$Compression\ Ratio : CR(m) = \frac{compressed\ size(m)}{original\ size(m)}$$

i.e., uncompressed data has a CR of 1.0. CR depends on both the lengths of the code symbols and the length (in symbols) of the input data. Our goal is to consume very little energy to achieve a low compression ratio for data from typical environmental sensor network applications.

Compression algorithms are either lossless or lossy. The former reduce the size of data to be transmitted whilst also maintaining accuracy, while the latter compress data within some error bound of the actual observed values. This paper focusses on lossless compression since accuracy is either required or at least can be considered as an upper bound for lossy algorithms. There have only been a few studies of lossless compression algorithms that are suitable for efficient execution on sensor network hardware and tailored to sensor network data.

Marcelloni and Veccio's simple lossless entropy compression algorithm (LEC) uses a generic Huffman code with 14 code words to encode the high order bits of each input symbol [7]. Then, each prefix code is concatenated with uncoded low order bits for the symbol. This approach has the advantage of a small codebook that is able to represent any 16-bit input symbol. LEC offers better compression ratios than dictionary-based algorithms such as S-LWZ [10] as well as having lower memory and energy use [7]. However, the generic code used in LEC does not perform as well as a pure Huffman code for difference data. This paper introduces an approach that gives better compression than LEC but has a similarly small codebook. An extensive evaluation of

real sensor data is used for examining the effects of data representation and codebook size independently, thus showing the contribution of each to successful compression.

Reinhardt et. al. [8] proposed an adaptive coding scheme for creating and adapting a small Huffman code that is updated as new symbols are seen. They analysed its performance on four data sets. The algorithm is "application-agnostic" in that the method is not tailored to any particular application. However, generic algorithms still have parameters that must be set, and we argue in this paper that data-awareness of the application leads to better compression ratios without any significant additional programming efforts for the designer.

S-LWZ is a dictionary-based, lossless compression algorithm for sensor networks based on Lempel-Ziv coding [10]. Sadler and Martonosi investigated the effect of block size, dictionary size, a mini-cache, and a dictionary overflow strategy on the performance of S-LWZ. However, both the compression ratio and performance costs of S-LWZ are higher than statistical coding algorithms such as LEC [7].

Some lossy compression algorithms can achieve lossless compression when their error bound is set to zero. For example, run length encoding is used by K-RLE, where runs of the same symbol, within an error bound K, are replaced by a (symbol,run-length)-pair [5]. In lightweight compression, sequences of microclimate sensor readings are replaced by a line segment approximation [12]. Latent variables can be used for estimating blocks of missing data [13]. However, although these algorithms have high compression ratios for non-zero error thresholds, they do not perform as well as statistical codes when lossless compression is required [7].

Algorithms designed for sensor network nodes must offer not only a high compression ratio, but also be able to run within the resource constraints of limited memory and processing power of sensor nodes. This constraint rules out a number of general-purpose compression algorithms, such as bzip [3], because of their relatively high memory and processing costs [10,1,7]. In fact, Barr and Asanović [1] have shown that using some of these algorithms can result in a net energy *increase* when compression is applied before transmission. And Reinhardt et. al. [9] showed that full adaptive Huffman coding may exceed the memory of sensor platforms, but performs no better than RLE which has minimal memory requirements. Another aspect of compression in sensor networks is fault tolerance to message loss. Guitton et. al. show that standard compression techniques may perform worse than no compression when a link has a greater than 10% packet loss rate [6].

Existing resource-aware compression algorithms designed for sensor networks focus on generic solutions, thereby excluding techniques that tailor the algorithm to the data to be compressed. In this paper, we propose and evaluate an algorithm that is both data-aware and resource-aware. Using experimental sensor data sets, we explain methods for determining the parameters of this algorithm based on analysis of the observations to be compressed.

The DARA algorithm is trained on a representative set of sensor readings together with user-provided constraints, such as target compression ratio, transmission block size, and memory limits. In the offline training stage, DARA produces, 1) a data transformation function and 2) codebooks for compression, as follows:

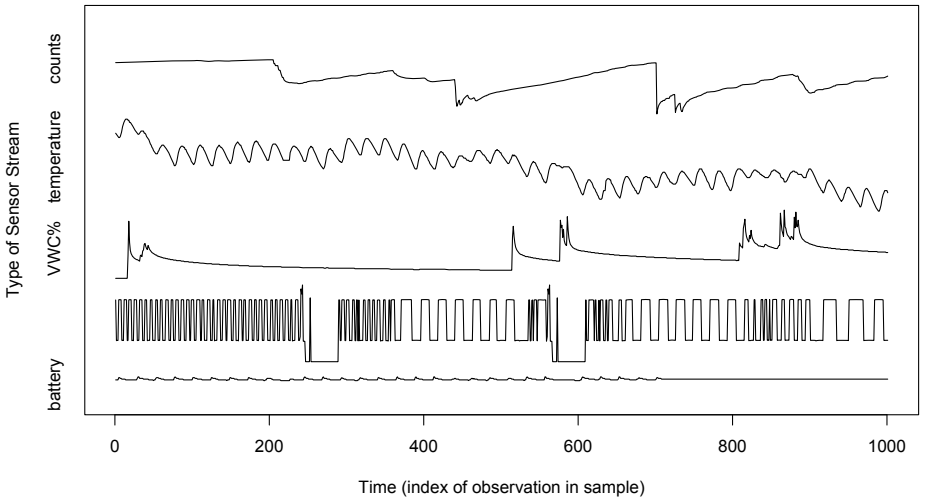


Fig. 1. Sample sensor data stream values from the benchmarks, 1000 observations each with y-axis scaled for readability

Stage 1: Data-Aware Transformation	
Given:	Training data as a matrix of raw sensor observations.
Generate:	A pipelined function to convert raw data into an equivalent but more compressible form.
Goal:	Minimize entropy and the number of symbols in the transformed data.

Stage 2: Resource-Aware Compression	
Given:	Transformed data from stage 1.
Generate:	Codebooks and block size for data compression.
Goal:	Minimize the size of transmitted data and codebook memory.

The DARA algorithm can be generalised for use with a number of different network models. For the evaluation in this paper, a single node was used with a phone modem sending daily digests by SMS to a nominated phone account. This node was directly connected to nine different sensors. DARA can also be used to determine compression parameters in networks where cluster head nodes receive sensor readings over the radio from their neighbours, and then transmit a compressed digest of all the readings to a base station.

3 Data-Aware Compression

In this section, we consider properties of sensor network data that can be exploited for compression and present an algorithm for pre-processing sensor data streams to optimise compressibility. The goal for a data-aware algorithm is to choose a representation for sensor data that maximises its compressibility without loss of accuracy.

Table 1. Sensor data benchmarks and their features

Data set ID	counts	hetero	calgeo
Number of Observations	14,116	24,408	5,000
Period of Observations	60 minutes	60 minutes	NA
Bits per observation	16	16	32
Entropy (bits per symbol)	11.92	7.32	12.05
Compression ratio CR	0.74	0.46	0.38
Number of symbols NS	5,973	300	4,618
Number of sensor data streams	4	9	1
Types of sensor data streams	frequency (4)	soil moisture (4) temperature (4) battery voltage (1)	bytes (4)
Range of sensor values	3.4e4 to 5.2e4	1.4 to 16.6 11.3 to 26.8 5.4 to 6.0	0 to 4.1e9

As a basis for analysis, we use a general model for sensor readings. Sensor data is represented by a matrix $M = [m_{i,j}]_{r \times c}$ containing symbols for observed sensor readings. Each of the r rows of the matrix contains readings from different sensors taken at the same time. A timestamp is associated with each row. Each of the c columns of the matrix contains consecutive readings from a particular sensor. The matrix can be heterogeneous: that is, different columns represent sensors that may sense different phenomena, such as soil moisture, soil temperature, humidity, sap flow, node battery, and so on.

3.1 Sensor Data Benchmarks

To explain and evaluate our algorithm, we have selected real-world, public domain benchmarks from long-running environmental WSN deployments [14] and a seismic data set [4]. The sensor data streams in these benchmarks include: soil moisture as frequency counts, soil moisture converted to scientific units (volumetric water content), soil temperature in degrees C, battery voltage, and 32-bit seismic observations. Figure 1 shows (scaled) samples of 1000 observations from each type of sensor in our benchmarks. This figure illustrates the wide variety of patterns seen in observed phenomena. Table 1 summarises the characteristics of each benchmark: its entropy (in average bits per symbol), from which we calculate the compression ratio that would be achieved using a perfect code. The *heterogeneous* data set contains soil moisture, temperature, and battery readings. The other sets are *homogeneous*, containing observations from only one type of sensor.

The following subsections introduce functions for transforming sensor data without loss of accuracy. These functions are pipelined together to transform raw data streams for compression. The choice of data representation is important because it can have a significant effect on the compressibility of the data.

Table 2. The counts benchmark as raw frequencies and converted to Volumetric Water Content (VWC%) with 0, 1, or 2 significant figures

Units	Significant Figures	Compression Ratio	Unique Symbols
frequency	0	0.74	5,973
VWC%	2	0.62	1,774
VWC%	**1**	**0.44**	**235**
VWC%	0	0.25	26

3.2 Engineering Units

A sensor reading, either digital or converted from an analog input, is typically represented by one or more bytes. For example, a positive integer represents a raw reading such as a dielectric soil moisture frequency, 16 bits are used for representing the sign and value of a temperature reading with one decimal place accuracy. Raw sensor readings are converted into engineering units using (for example) a polynomial conversion function. The result of the conversion is rounded to significant figures (SF) so that the measurements are reported to the accuracy of the sensing instrument that made the measurement. For example, in Table 1 each sensor data stream is represented accurately using one decimal place, i.e., 1 SF is appropriate for the accuracy of the sensors (shown in bold font), 0 SF gives lossy compression, and 2 SF is unjustified accuracy.

3.3 Temporal Correlation

Consecutive sensor readings from the same sensor are strongly correlated for most environmental phenomena. That is, the difference between a sensor reading and its temporal successor is a small value. A block of r rows of sensor readings can be represented by a timestamp followed by 1 row of values and $r-1$ rows of differences $d_{i,j} = m_{i+1,j} - m_{i,j}$ [7]. For example, a sample block of given soil moisture readings preceded by a timestamp from the counts benchmark is shown in the left hand box of Figure 2. The right hand box shows the same block represented by a time stamp in row 1 and an *anchor* of values (in VWC% engineering units) in row 2, followed by temporal *differences* for the remaining 3 rows of the block.

For most sensor data streams, using differences significantly improves compression ratios. However, the difference representation does not improve compressibility for the calgeo data since the periodic alternating pattern of that stream has a similar numbers and frequencies for the symbols in both its raw and temporal difference forms.

In order for a receiver to be able to recreate engineering values from an encoded difference matrix an *anchor row* of values $m_{1,1}, \ldots m_{1,c}$ together with a time stamp is transmitted with each block of difference data as shown in the example of Figure 2. Efficient methods for encoding both anchors and differences are discussed in Section 4.

3.4 Sensor Type and Sensor Location Correlations

The main distinction between the columns of a block of heterogeneous sensor readings is the different phenomena being measured by the sensor data streams. For example, the

2011-04-25 22:04			
24.5	32.5	36.4	33.4
24.4	32.5	36.4	33.3
24.4	32.5	36.4	33.3
24.4	32.5	36.4	33.3

2011-04-25 22:04			
24.5	32.5	36.4	33.4
-0.1	0.0	0.0	-0.1
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

Fig. 2. Absolute vs. difference representation of sensor data

hetero data set contains 3 types of data: soil moisture readings (in percentage volumetric water content); soil temperature (in degrees C); and battery voltage readings (in volts).

To take advantage of differences between different data types, we allow the data set D to be partitioned into subsets D_1 to D_k. The compression ratio for the whole data set is the *weighted sum* of the CRs for each subset. The total number of symbols that must now be coded is the *total sum* of symbols in each D_i. This step offers a trade-off between reducing the overall compression ratio and usually increasing the total number of symbols. The identity function is always one of the candidate splitting functions.

3.5 Data-Aware Compression Summary

Stage 1 of DARA generates a data transformation function that maps raw sensor data streams to an equivalent, but more compressible, form. The full process is summarised in Figure 3. Table 3 shows how the algorithm works out in practice for the benchmark data sets. Recommended representations are highlighted in bold in Table 3. Comparing rows Differences and Sensor Type, we see that for the hetero data set splitting on the type of sensor decreases the compression ratio from 0.15 to 0.11, *but* the number of symbols to be coded increases from 78 to 119. In short, the gain in compressibility may not justify the resource costs of learning and maintaining separate codebooks for a larger symbol set. For calgeo, the compression ratio actually increases from 0.38 to 0.51, but the number of symbols falls from 4530 to 46 when 32-bit code words are broken down into 4 x 8-bit codewords. Although the number of bits per symbol falls for each byte stream, the size of the data matrix is larger. In DARA, a weighted utility function is used for determining the best trade-off between compression ratio and number of symbols for transformed data given user targets for these values. Typically, the weight $\alpha = 0.5$, but this can be changed, depending on the application, to favour higher compression or reducing the symbol set.

4 Resource-Aware Compression

Stage 2 of DARA determines the parameters for compressing the transformed sensor data from Stage 1. The design goals for a resource-aware algorithm are to minimise the algorithm's memory use and execution time when the algorithm is run on sensor node hardware. The DARA compression algorithm is based on Huffman coding with modifications to address the resource constraints of sensor nodes.

Input: Training data $R = [m_{i,j}]_{r \times c}$ for the sensor network application
Output: Recommended transformation function $f = f_S \circ f_D \circ f_E$

Process:
$f_E(R) = E = [e_{i,j}]_{r \times c}$ with $e_{i,j}$ in engineering units ;
$f_D(E) = D = [d_{i,j}]_{r-1 \times c}$ with $d_{i,j} = e_{i+1,j} - e_{i,j}$;
do
TEC = ask user for target entropy compression ratio ;
TNS = ask user for target number of symbols ;
α = ask user for a weighting of *TEC* and *TNS* ;
TU = ask user for target utility value ;
F_S = ask user for a set of candidate split functions ;
u = *maxValue* ;
for each $f_{Si} \in F_S$ {
$\langle D_1, \ldots, D_k \rangle = f_{Si}(D)$; // get the split sub-matrices
for each $j \in [1, ..., k]$ { $w_j = |D_j|/|D|$; } // calculate weight of each sub-matrix
$cr = \Sigma_{j=1}^{k}\, CR(D_j) \times w_j$; // weighted compression ratio
$ns = \Sigma_{j=1}^{k}\, NS\,(D_j)$; // total number of symbols
$u_i = (\alpha \times cr/\,TEC + (1-\alpha) \times ns/\,TNS)$; // the utility value
if ($u_i \leq u$) { f_S=f_{Si} ; u=u_i ; } // select f_{Si} to minimize u
}
while ($u > TU$) ;
return $f = f_S \circ f_D \circ f_E$;

Fig. 3. Algorithm to generate a Data-Aware transformation function

4.1 Huffman Codes for Sensor Data

In order to encode an input stream efficiently, a compression algorithm requires a model for the symbols in that stream. Both context free models (e.g. in English text e is the most common letter and t is the next most common) and context sensitive models (e.g. q is usually followed by u) have been proposed, particularly for compressing text [2]. Huffman code is a context free algorithm. It maps input symbols to code words of different lengths based on the frequency of the input symbol, and it can achieve compression approaching the Shannon entropy limit.

A canonical Huffman code for the hetero benchmark requires a codebook with 300 symbols and codes of length 1 to 15 bits. Both differences and anchors are encoded. The advantage of a full Huffman code is that the compression ratio achieved can be equal to the best possible. However, this solution is not well suited for resource constrained sensor network nodes for a number of reasons:

- The codebook for 300 symbols is too large. Codebooks should be as small as possible on resource constrained nodes.
- The data has been over-fitted to the training data. Field data may contain different values that occur with different probabilities.
- New symbols that were not in the training set can not be encoded.

- Encoding will be sub-optimal if the frequencies in operational data differ from the training data.
- No allowance is made for missing data and erroneous sensor readings.

Stage 2 of DARA, described in the following sections, is an extension of the basic Huffman algorithm that is designed to address these problems.

Table 3. Data-Aware Analysis of the compression ratio (CR) and number of symbols (NS) for three benchmarks. The recommended representation for each benchmark is highlighted in bold for given targets and utility function.

Benchmark	counts			hetero			calgeo		
Utility $\alpha = 0.6$	TCR=0.25, TNS = 100			TCR=0.25, TNS = 100			TCR=0.60, TNS = 100		
Representation	CR	NS	Utility	CR	NS	Utility	CR	NS	Utility
Raw	0.74	5973	25.67	0.46	300	2.30	0.38	4618	18.85
Engineering	0.43	235	1.97	0.46	300	2.30	0.38	4618	18.85
Differences	**0.06**	**38**	**0.29**	**0.15**	**78**	**0.67**	0.38	4530	18.50
Sensor Type	0.06	79	0.46	0.11	119	0.74	**0.51**	**46**	**0.69**

4.2 Reducing the Codebook Size

The codebook for the hetero benchmark in engineering units has 300 entries: one for each different symbol in the training data. The codebook for hetero differences has 78 symbols. Even this may be too high an overhead for a memory resource-limited sensor node. A solution to this problem is to truncate the codebooks: use a Huffman code for only the most common input symbols and transmit the other symbols uncoded. The set of common symbols that are assigned Huffman codes is given by symbol set S, and a code symbol $\mathcal{U}$ is chosen to distinguish the uncoded symbols. For any symbol $s \notin S$, s is coded by $\mathcal{U}.s_b$. That is, the concatenation of the prefix $\mathcal{U}$ and symbol s restricted to b bits. A simple choice for $\mathcal{U}$ is to use the next Huffman code in the canonical sequence after assigning codes for the symbols in S. The number of bits b should be just large enough to encode all legal symbols: for example, 9 bits are sufficient for hetero differences.

Figure 4 (left) shows the relationship between codebook size and the compression ratio achieved using truncated codebooks. For the hetero benchmark, there is only a small penalty to pay in compression ratio by reducing the codebook size. For hetero differences, using a codebook with 20 symbols gives CR of 0.16, compared with CR of 0.15 for the full 78-symbol codebook. An alternative labelling strategy is to append a single bit to every codeword to distinguish between coded and uncoded values [8]. However, when symbols in S account for the majority of inputs, then our strategy requires fewer bits overall since the unknown symbol prefix will only rarely be used.

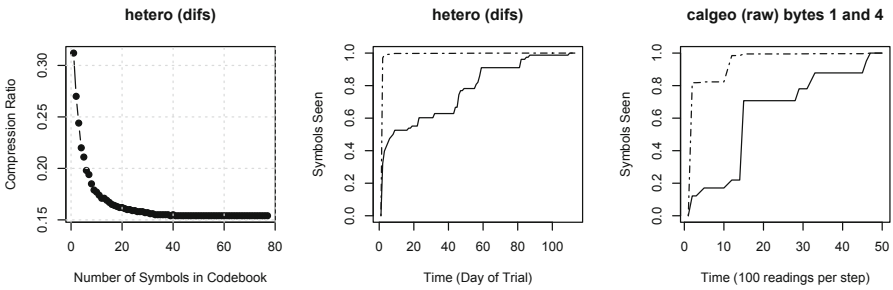


Fig. 4. Codebook size vs compression ratio for hetero differences (left), time versus proportion of symbols seen (solid line) and cumulative frequency of symbols (dotted line) for hetero differences (middle), and calgeo bytes (right) benchmarks.

4.3 Encoding New Observations

Another question is how to encode new symbols that are observed during operation of the sensor network, but that did not appear in the training data. This is known as the zero-frequency problem [2]. In DARA, new symbols are handled in the same way as rare symbols: each new symbol is encoded using the not-yet-coded prefix code. Figure 4 (middle and right) shows the proportion of different symbols seen versus time for the hetero and calgeo benchmarks. The results show a steady growth of new symbols seen over time. The dotted lines show the cumulative frequency distribution for number of symbols over time. After the first day (hetero) of 112 days, 97% of all sensor readings have already been seen, and 81% after the first 100 readings (calgeo) of 5000. Thus, although new symbols are seen later, such symbols occur with very low frequency and so can safely be treated in the same way as other rare symbols.

4.4 Missing and Erroneous Data

Sensor data is notoriously noisy: observed values may be out of range, out of context, or missing as a result of sensor or communication failure. We assume that erroneous observations can be identified by the sensor node and are reported using a unique symbol $\mathcal{E}$. When interpreting raw data blocks, $\mathcal{E}$ is replaced with an estimated value: the most recent in-range observation from the same sensor data stream. When calculating differences $\mathcal{E}$s have the same value as their predecessors, and so their difference value is 0. A special symbol $\mathcal{E}_0$ is used for estimated differences. If missing data occurs with high frequency then $\mathcal{E}_0$ may have its own symbol in the Huffman codebook. Otherwise $\mathcal{E}_0$ and $\mathcal{E}$ will be encoded using a prefix code in the same way as all other not-yet-coded symbols. The receiver can choose either to filter out estimated observations or to treat them as estimates for real observations.

4.5 Adaptive Coding for Changing Frequencies

When compressing a stream of sensor network data over time, the frequency of input symbols may change from their frequency in the training data. Frequency changes either occur because there is a change in the distribution of sampled data (for example, because of the time of the year in an environmental network) or because the original training data was not representative of the application data. Such changes lead to sub-optimal compression. Adaptive Huffman coding solves this problem.

In DARA, a simplified adaptive Huffman algorithm is used [11]. This algorithm does not guarantee an optimal tree. However, because of its simplicity it is better suited to implementation on resource constrained sensor nodes than the full adaptive Huffman algorithm used in other studies [8]. The sender and receiver follow the same process as follows. Start with a codebook created for the observed data. Both, transmitter and receiver store 3 columns: input symbol, a count of times the symbol has been seen, and the Huffman code for that symbol. The rows are ordered by the count column. If frequencies change, so that the ratio between symbol probabilities in rows i and $i+k$ is lower than the swap ratio threshold $SR \in [0,1]$, then swap the name and count of those rows, leaving the codewords as they were.

How does the receiver keep track of changes in the sender's codebook? If we have a feedback mechanism from the receiver to the sender, the receiver can require the sender to send a lost message again, and we have no problems at all (except for message delays). If we have no feedback mechanism from the receiver to the sender, the sender will not know which messages are lost. Now many strategies to treat this situation are possible, of which we prefer the following:

1. The sender can repeatedly send codebook changes after at most every N messages or, without extra costs, as part of each coded messages that, together with the codebook changes, does not exceed the allowed coded length.
2. After a message loss, the receiver can treat further received messages as being lost or depending on the thresholds, can treat some of these messages' values as being guaranteed and other values as being estimates only, until the receiver is again sure about the codebook.
3. To further reduce situations where the receiver is unsure about the codebook, the sender could enumerate its codebook versions and send a few bits with each block denoting the actual codebook version, such that the receiver can see whether or not the version is the same as his last codebook version even if a sequence of messages was lost.

In this sense, transmitted code blocks retain their property of being idempotent. That is, each message can be correctly decoded without reference to the preceding messages. The receiver can correctly adapt its codebook as soon as it receives a message from the sender even if a sequence of messages has been lost.

4.6 How to Encode Anchor Data

Huffman codes are inefficient when the probabilities of most input symbols are similar (rather than negative powers of two) and when there are a large number of symbols. For

example, when differences are used for representing data, then a known *anchor message* is needed in order to recover the original observed values. An example anchor message for the hetero benchmark is: `<12-06-25 16:00 3.5 8.2 8.7 6.8 15.6 15.0 15.2 14.8 5.6>` comprising a time stamp of 5 values followed by 9 sensor values. Anchor messages have a large number of symbols from different sensor data streams and the frequency distribution of those symbols is relatively uniform. Rather than using a Huffman code for anchors, we propose the following code:

1. Inputs are messages $< s_1, \ldots, s_n >$ where each symbol s_i may come from a different distribution.
2. For each field i, experts determine $scale_i$ to map symbol values to an integer range, min_i, max_i, and we calculate $range_i = max_i - min_i + 1$ for the scaled values. For example, possible battery values in the hetero benchmark are 3.5 to 6.5 Volts, giving $scale_i = 10$, $min_i = 35$, $max_i = 65$, and $range_i = 31$ for this application.
3. The number of bits required to represent field i is $b_i = \lceil log_2(range_i) \rceil$. For example, $b_i = 5$ bits are required to represent battery readings within a range of 31.
4. The code for symbol s_i is $int2bin((s_i \times scale_i) - min_i, b_i)$. For example, battery value 5.7 is represented by (57 - 35) = 22, in 5-bit binary as 10110. Of course, the scaling and the range computation have to consider the accuracy of the sensed values, i.e., if the battery accuracy is 0.05 Volts with allowed values from 3.5 to 6.5 Volts, we have to represents 61 possible values, needing a 6-bit code.
5. The anchor codebook consists of three n-vectors: $scale$, min and b.

Anchors encoded in this way for the hetero benchmark require 104 bits per anchor which is 7.4 bits per symbol (b/s). This is slightly higher than the entropy of 6.9 b/s for this data, but has the advantage of a very small codebook of 3 by 14 entries compared to a Huffman code for over 300 unique anchor symbols. Anchor coding is, however, still much less efficient than Huffman coding of differences for the remainder of the data where the CR is 0.16.

4.7 Resource-Aware Compression Summary

Figure 5 summarises the resource-aware steps of the DARA algorithm. The overall compression ratio achieved is a combination of the ratios for the anchor and Huffman codes. This target compression ratio will be slightly higher than the entropy target used in Stage 1 of the algorithm. A block size for transmitted messages is chosen to achieve the target compression ratio and satisfy codebook memory constraints.

5 Evaluation: An SMS Sensor Network

In order to evaluate the practicality of DARA, we implemented a sensor network that reports its data using SMS messages. A mobile phone sensor node was built and deployed using: a high-performance, low power microcontroller (ATmega 1284p), GSM GPRS quad-band module (TELIT GC 864-QUAD); 4 sensor ports; on-board temperature and battery sensor; real time clock; SD card; and serial flash. The node was powered by rechargeable batteries and a solar panel. The whole application, including encoding,

Input: Training data $R = [m_{i,j}]_{r \times c}$ for the sensor network application
Output: Recommended codebooks $C_1, \ldots, C_k, A$ and block size B

Process:
$f = f_S \circ f_D \circ f_E$; // calculated for R as specified in Figure 3
$\langle D_1, \ldots, D_k \rangle = f(R)$; // transform data for Huffman encoding
for each $i \in [1, ..., k]$ {
  $w_i = |D_i|/|D|$; //calculate weight of each sub-matrix
  $H_i = HuffmanCodebook(D_i)$; // calculate a full Huffman codebook for each D_i
}
$f_E(R) = E = [e_{i,j}]_{r \times c}$ with $e_{i,j}$ in engineering units for anchor rows ;
$A = AnchorCodebook(E)$; // calculate anchor codebook
crA = *CR*$(A(E))$; // calculate compression ratio for anchor encoding
TU = ask user for target utility value ;
u = *maxValue* ;
do {
  TCR = ask user for target compression ratio ;
  TCS = ask user for target total codebook size ;
  B = ask user for target transmission block size ;
  α = ask user for a weighting of *TCR* and *TCS* ;
  $\langle n_1, \ldots, n_k \rangle$ = ask user for truncated codebook sizes with $n_j \leq |C_j|$;
  for each $j \in [1, ..., k]$ {
    C_j =*truncate*(H_j, n_j) ; // truncate each full codebook to n_j entries
  }
  crD= $\Sigma_{j=1}^{k}$ *CR*$(C_j(D_j)) \times w_j$; // weighted compression ratio
  cr = $(1 \times$ *crA* + $(B-1) \times$ *crD*$)/B$; // overall compression ratio
  $ns = |A| + \Sigma_{j=1}^{k} |C_j|$; // total size of all codebooks
  $u_c = (\alpha \times cr/TCR + (1-\alpha) \times ns/TCS)$; // the current utility value
  if ($u_c \leq u$) { $u = u_c$; } // select codebooks to minimize u
}
while ($u > TU$) ;
return $C_1, \ldots, C_k,\ A,\ B$;

Fig. 5. Algorithm to generate Resource-Aware compression codebooks

uses 19.0% of the program memory and 43.8% of the data memory, easily meeting the requirement of resource-awareness for sensor node hardware since this SMS application has no requirements for routing or other network management code. Our evaluation used 9 sensor data streams: 4 soil moisture, 4 soil temperature, and 1 battery voltage sensor. Readings were taken once per hour, and reported in an SMS message once every 24 hours. Additional messages to report alarm conditions outside this schedule could also be sent as required, but are not part of this evaluation.

SMS messages are written using a subset of the 128 ASCII symbols. However, there are some differences in the SMS codes used by different manufacturers since selected ASCII symbols are actually transmitted as 2 characters. So we restrict our code to 64 non-controversial character symbols each of which can be represented by a 6-bit binary string. The binary Huffman encoding of a coded block produced by DARA is mapped to an SMS transmittable stream by replacing 6 bits at a time with one of the selected

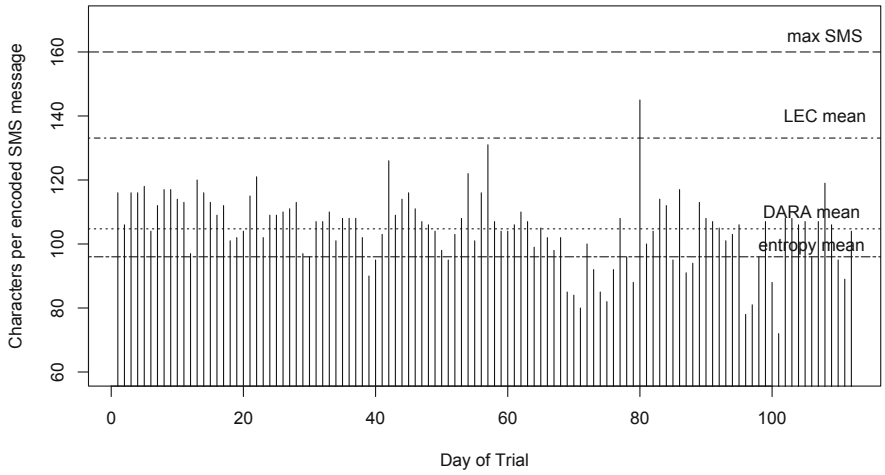


Fig. 6. Code length in characters of each encoded message for evaluation data set. Horizontal lines show the mean message lengths for DARA and related encodings.

ASCII SMS symbols. In this way, a 160 character SMS message can encode a block of up to 960 coded bits.

Using the DARA algorithm, a single codebook for differences was defined with 20 entries, block size $B = 24$ hours, and an anchor codebook. Figure 6 shows the length of the resulting SMS encoded message (in characters) for each day of the evaluation. The mean message length was 104.7 characters which is a compression ratio of $CR = 0.24$. The shortest encoded message had 72 characters and the longest 145, i.e., no messages in the evaluation were over the 160 character limit for SMS messages (shown as max SMS). We also encoded the evaluation data using the LEC compression algorithm [7]. The four horizontal lines in Figure 6 compare the mean length of DARA messages with the entropy mean, the LEC mean and the maximum SMS message length. The mean message length for the DARA algorithm (shown as DARA mean of 104.7) is close to the entropy lower limit (shown as entropy mean of 96). The mean compression ratio for LEC (shown as LEC mean of 133.1) is 28.6 characters longer than the mean DARA. The standard deviation of DARA was 11.1 characters and for LEC is 7.7 characters. Both algorithms have similar memory and execution requirements.

According to their data sheets the TELIT GC 864-QUAD typically draws <420 mA for GPRS transmissions while the ATmega 1284p draws 0.4 mA in active mode. So, in practice, no more than 1% of the energy is used for sensing and compression, and 99% for sending the SMS message. Without compression, our system could send at most 2 sets of readings in a single SMS message, using a format such as Table 2 (left). This would require 12 SMS messages to be sent per day. With compression, a whole day of readings can be compressed into a single SMS message. Given the dominating energy cost of transmissions, saving 11 SMS messages in each a sequence of 12 packages sent to the receiver, is worth much more than the best possible energy-aware optimisations of a compression algorithm.

Since we assumed that there is no message loss in our experimental SMS scenario, we could avoid spending a few extra bits for sending codebook versions and repeatedly sending codebook changes after at most every N messages, although we still transmitted an anchor with each message. Within the evaluation period, there were no codebook changes required. So, for this case, there was no overhead for codebook maintenance. Where codebook maintenance is required, there were, on average, 55 characters available per message, and so the codebook maintenance information together with the compressed sensed data would easily fit into a single SMS message. In summary, DARA comfortably meets its compression requirements for this application, having a mean message length close to Shannon entropy limit and being 28 characters per message shorter than LEC, which has the best performance of existing sensor network compression algorithms. Further, LEC transmits only encoded differences with no anchors, and so it relies on the assumption that no messages will be lost in transmission.

6 Conclusions

This paper presents the design and evaluation of a generic algorithm, DARA, for efficient compression of sensor network data. Previous studies have focussed on resource-awareness, optimising existing algorithms to respect resource constraints. Our approach is both data-aware and resource-aware. The DARA algorithm defines a data-aware mapping from raw sensor readings to an equivalent form that maximises compression. Once this is done, standard Huffman coding methods can be used, albeit with some modifications for resource constrained nodes. The DARA algorithm is general purpose: it is suitable for heterogeneous or homogeneous sensor data streams; it can be trained on different types of sensor data; and it can be configured to achieve higher or lower compression ratios as demanded by different applications. The algorithms is also adaptive: coding parameters are changed on-the-fly when necessary. In comparison with existing compression algorithms for sensor networks, the advantage of our algorithm is that it can easily integrate expert knowledge about the optimal data representation, which leads to better compression ratios whilst still respecting the resource constraints of sensor nodes. In particular, practical use of DARA is demonstrated by an SMS application that requires blocks of sensor readings to be compressed to one quarter of their original size.

There are a number of possible directions for future work. In considering an SMS application, the problem of lost messages was insignificant. However, if the compression algorithm were to be used in a less reliable communication environment, then further study of methods for recovering lost messages and managing lost state information for adaptive coding would be needed. Further investigation of the effect of adaptive coding in long running applications is needed, and comparison with existing data stream-mining methods and new types of sensor data streams.

Acknowledgements. This work was partially supported by the Go8-DAAD Australia-Germany Joint Research Co-Operation Scheme under grant DAAD10505012.

References

1. Barr, K.C., Asanović, K.: Energy-aware lossless data compression. ACM Trans. Comput. Syst. 24(3), 250–291 (2006)
2. Bell, T., Witten, I.H., Cleary, J.G.: Modeling for text compression. ACM Comput. Surv. 21(4), 557–591 (1989)
3. bzip2 (2012), http://bzip.org (retrieved January 2012)
4. Calgary Corpus Geophysical data (1987), www.data-compression.info/Corpora/CalgaryCorpus/ (retrieved June 2012)
5. Capo-Chichi, E.P., Guyennet, H., Friedt, J.-M.: K-RLE: A new data compression algorithm for wireless sensor network. In: Third International Conference on Sensor Technologies and Applications, SENSORCOMM 2009, pp. 502–507 (June 2009)
6. Guitton, A., Trigoni, N., Helmer, S.: Fault-Tolerant Compression Algorithms for Delay-Sensitive Sensor Networks with Unreliable Links. In: Nikoletseas, S.E., Chlebus, B.S., Johnson, D.B., Krishnamachari, B. (eds.) DCOSS 2008. LNCS, vol. 5067, pp. 190–203. Springer, Heidelberg (2008)
7. Marcelloni, F., Vecchio, M.: An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks. Computer Journal 52(8), 969–987 (2009)
8. Reinhardt, A., Christin, D., Hollick, M., Schmitt, J., Mogre, P.S., Steinmetz, R.: Trimming the Tree: Tailoring Adaptive Huffman Coding to Wireless Sensor Networks. In: Silva, J.S., Krishnamachari, B., Boavida, F. (eds.) EWSN 2010. LNCS, vol. 5970, pp. 33–48. Springer, Heidelberg (2010)
9. Reinhardt, A., Christin, D., Hollick, M., Steinmetz, R.: On the energy efficiency of lossless data compression in wireless sensor networks. In: IEEE 34th Conference on Local Computer Networks. LCN 2009, pp. 873–880 (October 2009)
10. Sadler, C.M., Martonosi, M.: Data compression algorithms for energy-constrained devices in delay tolerant networks. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys 2006, pp. 265–278. ACM, New York (2006)
11. Salmon, D.: Huffman Coding. In: A Concise Introduction to Data Compression, ch. 2, Springer, Heidelberg (2008)
12. Schoellhammer, T., Greenstein, B., Osterweil, E., Wimbrow, M., Estrin, D.: Lightweight temporal compression of microclimate datasets. In: 29th Annual IEEE International Conference on Local Computer Networks, pp. 516–524 (November 2004)
13. Verma, N., Zappi, P., Rosing, T.: Latent variables based data estimation for sensing applications. In: 2011 Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2011), Adelaide, Australia, pp. 335–340 (December 2011)
14. WebSense: Sensor Network Viewer (2011), wsn.csse.uwa.edu.au/ (retrieved June 2012)

BLITZ: Wireless Link Quality Estimation in the Dark

Michael Spuhler[1], Vincent Lenders[2], and Domenico Giustiniano[1]

[1] ETH Zürich, Switzerland
[2] Armasuisse, Switzerland

Abstract. We present BLITZ, a novel link quality estimator that relies on physical-layer synchronization errors to estimate the expected packet delivery ratio of wireless links. In contrast to all existing link quality estimators which estimate the packet delivery based on statistics from packets that are *successfully* decoded, our technique works even when packets at the receiver are *not correctly received*, i.e., when the synchronization fails. The core idea of BLITZ is to exploit information from chip errors in the received preamble of any transmitted direct sequence spread spectrum signals such as IEEE 802.15.4. Using extensive measurements over cable, wireless static and wireless mobile scenarios, we show that our proposed estimator outperforms existing estimators in terms of both accuracy and speed. Across diverse environmental conditions and the full range of possible link qualities, BLITZ provides packet delivery ratio estimates with an absolute error below six percent within just a few milliseconds.

1 Introduction

A fundamental problem of wireless network protocols consists of link quality estimation. Link quality estimation is a crucial building block for higher layer protocols in wireless networks. The performance of routing, rate selection, handover, jamming detection, or network coding heavily depends on accurate and fast wireless link quality estimator. For example, long prediction time for rate selection would rapidly deteriorate the link quality before action has been taken. Despite these needs, the task of estimating the effective link quality in real-life wireless networks remains a challenge. Particularly in dynamic link environments, where moving nodes foster the unpredictable and location-sensitive nature of wireless channels, estimating the packet delivery probability *efficiently*, *quickly* and *accurately* proves to be a difficult task.

Current wireless link quality estimators generally fail to provide accuracy, reactivity and stability at the same time across diverse environments and conditions. For example, existing SNR [1] and chip errors [2] based estimators tend to be fast but empirical studies have shown that they often fail to accurately predict the packet delivery ratio (PDR) [2–5]. Estimators that rely on historical packet count statistics [6–8] provide a more accurate representation of the channel conditions. However, these models are relatively slow as they require a few packets for building up meaningful packet statistics (typically more than ten packets). Hybrid models combining the above approaches [9–11] have shown to further improve the accuracy. Yet, the estimation time remains high as a few packets are still required to build up a reliable link estimation.

A common limitation of all current link quality estimators is that they update statistics exclusively when packets are successfully transmitted or received. We argue in this

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 99–114, 2013.

paper that this approach has two fundamental drawbacks. First, in existing SNR and chip errors-based estimators, the measured metrics are biased towards an estimation of good channel conditions since only successful packets are considered. Second, and most important, in all current link quality estimators, low quality links that exhibit a high number of packet losses require considerable amounts of time to be sampled. For example, a link with a low packet delivery ratio of 20% requires statistically the transmission of 5 packets to obtain one successful packet for actual estimation.

This paper proposes BLITZ, a novel estimator of the packet delivery ratio for point-to-point communication that analyzes errors in *frame synchronization*. In contrast to all existing estimators, BLITZ does not require any successful packet transmissions to estimate the link quality as it is able to analyze the synchronization of packets even when packets are lost[1]. BLITZ operates on direct sequence spread spectrum (DSSS) systems in which each information symbol is mapped to multiple chips. The key idea is to analyze chip error patterns in preamble symbols during the synchronization phase. Given this mapping of one symbol to many chips, BLITZ relies on a rich source of information to estimate the channel state over a short duration. For example in IEEE 802.15.4, a popular sensor network standard that relies on DSSS, the preamble consists of 8 symbols of 32 chips each, resulting in 256 chips per preamble that can be used to estimate the channel state in $128\,\mu s$.

We have implemented BLITZ on the USRP software-defined radio platform and tested its performance under idealized and real-world wireless channel conditions. In particular, we have compared its performance to existing estimators that rely on packet statistics, SNR, and chip errors in the payload. Our results show that BLITZ is superior to all estimators in terms of both accuracy and reactivity. In particular, we show that

- BLITZ has the best performance over all wireless scenarios with an average PDR estimation error of $5.7\,\%$,
- because of its fast estimation, BLITZ is particularly suitable in mobile settings where it outperforms on average all other estimators by $10\,\%$ to $20\,\%$ with regard to the absolute PDR error,
- because BLITZ does not rely on successful transmissions to estimate the link quality, the PDR is estimated accurately within just a few milliseconds, and regardless of the (good or bad) link quality.

To the best of our knowledge, this work is the first to show that errors in the synchronization phase of wireless frame transmissions can be used to accurately model the packet delivery over wireless links.

The rest of this paper is structured as follows. In the next section, we present relevant background information on the IEEE 802.15.4 standard which is the base for our analysis. Section 3 analyzes the characteristics of wireless synchronization errors in different environmental conditions. Section 4 presents BLITZ, our novel link quality estimator. The performance of BLITZ is compared to other estimators in Section 5. Section 7 discusses related work and Section 8 finally concludes the paper.

[1] We refer to this capability as link quality estimation in the **dark** because traditional estimators do not see packets that are lost due to synchronization failures.

2 Background on IEEE 802.15.4

Our work on link quality estimation focuses on direct sequence spread spectrum (DSSS) communication systems and in particular on the IEEE 802.15.4 standard [12]. This section gives a brief overview of DSSS and how it is employed in IEEE 802.15.4.

2.1 Modulation and Decoding

The IEEE 802.15.4 standard defines a 16-ary quasi-orthogonal modulation technique based on DSSS. This modulation spreads a low rate sequence of bits to a higher rate sequence of so called chips. The binary source data is divided into groups of 4 bits (referred to as symbols) and mapped to a nearly orthogonal 32-chip pseudo-noise sequence $(b_0, b_1, b_2, b_3) \mapsto (c_0, c_1, \ldots, c_{31})$, resulting in a chip rate of 2 MChips/s.

A frame in IEEE 802.15.4 consists of a preamble sequence (eight 0 symbols), a start of frame delimiter (symbol 7 followed by symbol 10), a frame length field and a MAC protocol data unit (MPDU). The MAC protocol data unit ends with a frame check sequence (FCS) which is used to detect errors in the MAC payload.

At the receiver side, the signal is decoded using a correlator to map the received 32-chip sequences back to symbols. When the receiver detects a signal on the channel, it synchronizes on the frame by checking the incoming chip sequences and comparing them to the expected sequences of the preamble and start of frame delimiter. The received chips may contain errors caused by fading or interference. The received chip sequence R is interpreted correctly as C, when the hamming distance is smaller than a threshold: $h(R, C) < \text{threshold}$, where $h(\cdot, \cdot)$ is the hamming distance (number of positions containing different chips) between the two arguments. For example $h((01101), (01110)) = 2$ because the two vectors vary in the last two bits which are flipped. After successful synchronization, the following chip sequences are decoded according to the best match, i.e., the received chip sequence R is compared to the 16 predefined chip sequences $C_i, i = 1, 2, \ldots, 16$. The receiver chooses the best match, i.e. the C_i, such that $h(R, C_i)$ is minimized. If too many chips are flipped, the expression $h(R, C_i)$ may be minimized for the wrong chip sequence C_i, such that the receiver interprets the received chip sequence as a wrong symbol.

2.2 Synchronization and Packet Losses

The preamble sequence and the start of frame delimiter are the basis of the synchronization process between a transmitter and a receiver in IEEE 802.15.4. Figure 1 illustrates schematically the synchronization phase of two packets, where the first packet is lost due to a synchronization error and the second packet is transmitted successfully. In (a), the sender starts to transmit the preamble sequence, the SFD (start of frame delimiter) and the corresponding part of the packet (named here as the rest of packet). During the transmission of the eight preamble symbols of the first packet, $P_{1,2}$, $P_{1,3}$, $P_{1,4}$ could not be decoded correctly due to high number of chip errors. E.g. $P_{1,7}$ was transmitted successfully because as shown in (d) only three chips were flipped during the transmission, and the maximum error threshold to discriminate between a correct or wrong preamble symbol is not exceeded. Due to a corrupted symbol in the SFD_1 the synchronization of

the first packet fails and the receiver is not able to decode this entire packet. Contrary to the first packet, the second packet is transmitted successfully (c) and only the preambles $P_{2,1}$ and $P_{2,5}$ were not correctly decoded.

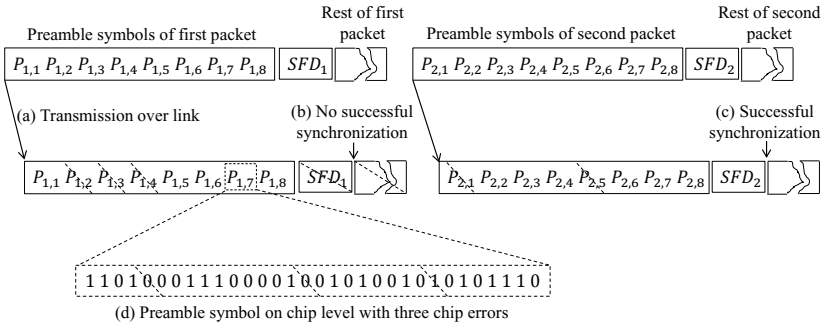


Fig. 1. Examples of how chip errors in the preamble affect packet losses

3 Analysis of Packet Synchronization Errors in IEEE 802.15.4

This section analyzes the packet synchronization errors in IEEE 802.15.4. Our analysis focuses on environments that are free of interference. First, we show that in such environments most packet errors in IEEE 802.15.4 are due to synchronization failures while packet errors caused by symbol errors in the payload are rare. Then, our analysis deepens on two key error patterns in synchronization: (i) the number of preamble symbols that are wrong at the receiver and (ii) the number of chip errors in the received symbols of the preamble. The insights of this analysis will serve as the basis for the development of our novel estimator BLITZ in the next section.

3.1 Experimental Setup

For our analysis, we conduct experiments with a software-based implementation of IEEE 802.15.4. As hardware platform, we use the USRP software-defined radio from Ettus Research. For the software, we use a slightly optimized version of the UCLA IEEE 802.15.4 implementation [13] that runs on the GNU Radio framework. We perform multiple tests in various indoor environments which are summarized as *cable*, *static* and *mobile* experiments. In the *cable* experiments, sender and receiver are connected by a shielded 60 cm coaxial cable with a 30 dB attenuator. The *static* experiments correspond to scenarios in which a stationary sender and receiver communicate over omni-directional antennas. The *mobile* experiments are similar to the static scenarios except that the sender is kept stationary while the receiver is moving. The receiver is placed on a cart and moved at a constant speed of maximum $v = 1$ cm/s away from, and back towards, the sender.

In each experiment run, 40,000 packets of 26 bytes length are sent during 40 seconds from the transmitter to the receiver at constant bit rate. Various link conditions in the

cable and static experiment runs are obtained by adjusting the transmit power and by changing the position of nodes respectively. The true packet delivery ratio (PDR) of a link at time t is calculated by averaging the number of received packets over a window of 100 packets centered around t. A window size of 100 packets assures that the true PDR is calculated over a time window which is smaller than the channel coherence time[2] when moving the receiver at maximum $v = 1$ cm/s and at a frequency of 2.4 GHz. Note that the mobility experiments have a relatively low node speed of maximum 1 cm/s for the sake of determining the true PDR. We intentionally kept the node mobility low such that the channel coherence time is larger than the window size of 100 packets that are used to calculate the true PDR. Our results are hence relatively conservative with regard to the mobility.

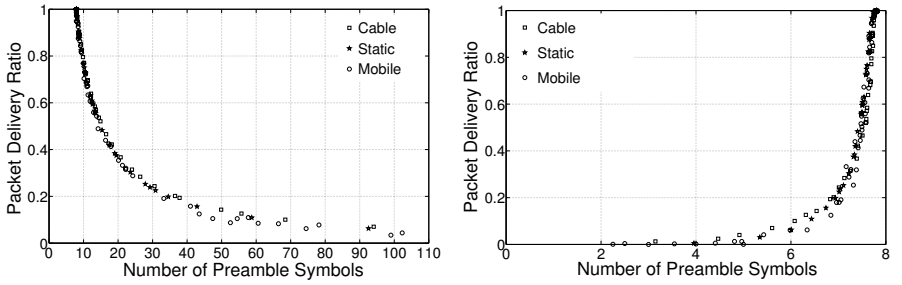
3.2 Analysis

Root Cause of Packet Errors. There are two reasons why packets in wireless networks, and in particular in IEEE 802.15.4 networks, may get lost: (i) The receiver is not able to synchronize on the preamble and start of frame delimiter of the frame and misses the entire frame or (ii) the receiver is able to synchronize on the frame but decoding errors occur in the received symbols of the payload such that the frame check sequence (FCS) is incorrect and the frame is discarded. Our experiments clearly show that in the absence of interference, the dominating cause of packet errors in IEEE 802.15.4 is the synchronization failure. Figure 2 shows experimentally the probability that a packet is lost because of synchronization failures. This probability is larger than 99.7% across the entire range of PDR values. Since synchronization failures are dominating, the best place to analyze packet errors is therefore at the synchronization phase in the preamble. Note that larger packet sizes may increase the probability of packet losses due to erroneous frame check sequences, however even for the maximum packet size of 127 bytes, missed packets caused by synchronization failures remain the dominating root cause of errors.

Fig. 2. Probability of a synchronization failure given that the packet is lost

[2] The coherence time is the time duration over which the channel impulse response is considered to be not varying and is approximately $\frac{1}{4D}$, where D is the Doppler spread.

Symbol Error Pattern in the Preamble. Looking at errors in the preamble, we analyze first how many preamble symbols a receiver misses/receives on average before it successfully receives a packet. Figure 3(a) shows the distribution of the PDR versus the average number of preamble symbols that a receiver decodes before synchronizing on a frame for the cable, static and mobile scenarios. We see that the average number of decoded preamble symbols is not affected by the type of environment and link conditions. Another observation is that the average number of decoded preamble symbols can be much larger than eight (the fixed number of preamble symbols in the IEEE 802.15.4 standard) for low quality links. For example, an average of 15 preamble symbols is obtained for a PDR $= 50\%$ while it increases up to more than 100 when the PDR drops below 10%. The reason is that the receiver decodes preamble symbols from multiple packets before eventually synchronizing successfully on a frame. Overall, the distribution is monotonic and the average number of decoded preamble symbols per received packet is clearly an indicator of the link quality. However, the correlation is not conclusive. For example, the Pearson product-moment correlation coefficient which provides a measure of the strength of linear dependence between two variables as a value between $+1$ and -1 is here $r = -0.5585$, which indicates a relatively low linear correlation between the two metrics[3].



(a) Average number of decoded preamble symbols per successfully delivered packet.

(b) Average number of consecutively decoded preamble symbols per transmitted packet.

Fig. 3. PDR versus the average number of received preamble symbols (a) per successfully decoded packet and (b) per sent packet

In a second step, we analyze the number of consecutively decoded preamble symbols per transmitted packets. An average of this number versus the PDR is visualized in Fig. 3(b). Again, the distribution is unaffected by the type of scenario. The Pearson correlation coefficient is here $r = 0.7617$ indicating a higher linear correlation for the average number of consecutively decoded preamble symbols, but for all links with a PDR above 20%, the average number of decoded symbols lies in a narrow, not strictly well distinctive range of values between 7 and 8.

[3] Values close to 0 refer to a low correlation while values close to $+1$ and -1 represent a high correlation.

Chip Error Pattern in the Preamble. Figure 4 shows the distribution of chip errors per preamble symbol versus the PDR. As before, the environmental conditions do not affect the distribution. The Pearson correlation coefficient is $r = -0.9650$, showing the highest correlation for symbol error patterns in the preamble.

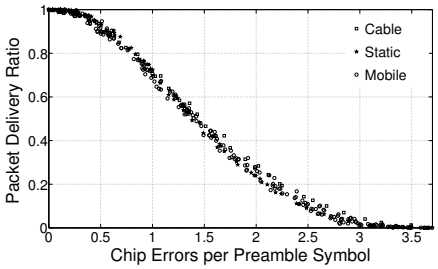


Fig. 4. Correlation of average chip errors per preamble symbol at the receiver and PDR

Summary of Results. In summary, we conclude that the dominating root cause of packet errors in IEEE 802.15.4 are synchronization failures. Looking at synchronization failures is therefore well suited to observe packet error behavior. Across cable, wireless static and mobile settings, the number of missed symbols and the number of chip errors in the synchronization phase both correlate well with the PDR. However, the correlation of missed preamble symbols to the PDR is not as distinctive and hence not well suited to model the PDR. In contrast, the average number of chip errors in the preamble has a high linear correlation to the PDR. To model the link quality with BLITZ, we therefore exclusively rely on chip errors in the preamble symbols.

4 BLITZ: Preamble-Based Link Quality Estimation

This section presents BLITZ, our novel link quality estimator that relies on chip errors in the preamble to estimate the PDR of wireless links in point-to-point communication. To meet the requirements of accuracy and stability of the quality estimation, BLITZ operates at two-time scales. At the preamble level, chip errors of received symbols are first averaged and fitted to a polynomial model to obtain an estimation of the instantaneous PDR. At the packet level, chip error statistics from multiple transmitted packets are filtered according to a weighted moving average function to smooth short-term fluctuations of the estimation method.

4.1 Instantaneous PDR

In a first step, BLITZ estimates an instantaneous PDR after the reception of the preamble of packet k as

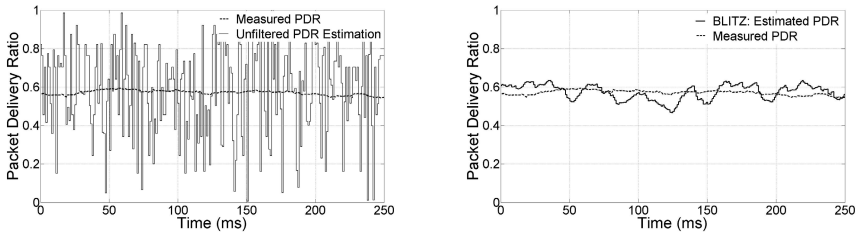
$$PDR_{\text{inst}}(k) = g\left(\frac{\sum_{i=1}^{32}\sum_{j=1}^{|\mathcal{S}_k|}\left(P_{k,j}[i] \oplus P[i]\right)}{|\mathcal{S}_k|}\right),$$

where $P_{k,j}[i]$ is a vector containing the 32 chips of the j-th received preamble symbol of packet k for $i = 1, 2, \ldots, 32$, $P[i]$ denotes a vector with the correct chips of the known preamble symbol, $\oplus$ is the exclusive or operator, and $|\mathcal{S}_k|$ is the number of received preamble symbols for packet k. The function $g(\cdot)$ models the empirical distribution of the PDR versus chips errors per preamble symbols as shown in Figure 4. In this work, we fit a polynomial of the 5-th degree to the distribution according to

$$g(p) = \mathrm{p}_0\, p^5 + \mathrm{p}_1\, p^4 + \mathrm{p}_2\, p^3 + \mathrm{p}_3\, p^2 + \mathrm{p}_4\, p + \mathrm{p}_5$$

while minimizing the root square of the error. The obtained parameters of the fit are $p_0 = 0.016, p_1 = -0.33, p_2 = 2.41, p_3 = -7.26, p_4 = 8.83, p_5 = -3.24$ resulting in a root mean square error below 3% across the entire distribution[4].

While the instantaneous PDR provides a very fast estimation of the link quality, it is subject to high fluctuations as shown in Figure 5(a). In order to provide a more stable link quality metric, BLITZ further averages and filters consecutive instantaneous PDR estimations as described next.



(a) Instantaneous PDR: high fluctuations.

(b) Filtered PDR: low fluctuations.

Fig. 5. BLITZ: Fluctuation of the instantaneous (left) and filtered (right) PDR estimation on a wireless static link with a true PDR of approximately 55%

4.2 Averaged and Filtered PDR

A classical approach to increase the stability of a metric is to weight sequential estimations in form of a weighted moving average. For example, the estimator in [6] makes use of this technique to increase the stability of packet count statistics estimators. BLITZ applies a similar approach in order to smooth consecutive estimations of the instantaneous PDR. For this reason, BLITZ performs a low-pass filtering of the weighted average over a window w of consecutive link estimations. Suppose $\mathcal{PDR}_{\text{inst}}(k)$ is the

[4] Note that we use a polynomial of the 5-th degree to understand the best performance that an estimator like BLITZ may achieve. A polynomial of a smaller degree may alternatively be used in real-world deployments at the cost of a marginal error increase since the underlying empirical distribution is close to linear.

set of the past $l+1$ estimations of the instantaneous PDR at the position k. Let further $PDR_{\text{inst}}(k-l), PDR_{\text{inst}}(k-l+1), \ldots, PDR_{\text{inst}}(k-1), PDR_{\text{inst}}(k) \in \mathcal{PDR}_{\text{inst}}(k)$ be the past $l+1$ estimations. Then, the weighted average $\text{wa}(k)$ over these recent $l+1$ estimations at the position k is calculated as

$$\text{wa}(k) = \sum_{m=0}^{l} \beta_m PDR_{\text{inst}}(k-m),$$

with the weighting factors β_m such that $\sum_{m=0}^{l} \beta_m = 1$. Using this weighted average $\text{wa}(k)$, we compute the output of the low pass filter $f_{\text{wa}}(k)$ (filtered weighted average) as

$$f_{\text{wa}}(k) = \alpha\, f_{\text{wa}}(k-1) + (1-\alpha)\left(\frac{1}{\text{wa}(k)} - 1\right),$$

where $\alpha \in [0,1]$ controls the smoothness. For example, a small factor α gives more importance to current link behavior. Finally the k-th estimation is obtained as

$$PDR(k) = \frac{1}{1+f_{\text{wa}}(k)}.$$

The benefits of the averaging and filtering are illustrated in Figure 5 (b). The figure shows the resulting estimation error after filtering and weighting the estimations. With the parameters l, β_m, and α, the estimation window can be changed to tune the reactivity of the estimator. In this work we set $l = 6$, $\beta_0 = 0.3, \beta_1 = 0.2, \beta_{2,\ldots,6} = 0.1$, and $\alpha = 0.9$ as it provides the best trade-off in our experiments.

5 Evaluation and Comparison to Existing Link Quality Estimators

In this section, we show how BLITZ performs under different link conditions. To assess its performance, we compare BLITZ to competitive estimators from the literature. We consider a wide class of estimators ranging from estimators that rely on packet count statistics, signal strength, and chip errors in the payload.

5.1 Considered Estimators

We compare the performance of BLITZ with five estimators:

ETX [7]: ETX is an estimator that relies on packet count statistics. By sending broadcast probes at an average period τ within a window of w seconds, each node can estimate the probe reception rate. The PDR is calculated in one direction as $PDR_{\text{ETX}} = \frac{\text{probes}(t-w,t)}{w/\tau}$, where $\text{probes}(t-w,t)$ is the number of probe packets received during the window w, and w/τ is the number of probes that should have been received. We set $\tau = 1$ second and $w = 10$ seconds as proposed in [7].

WMEWMA [6]: Like BLITZ, this estimator uses low pass filtering and calculates a weighted moving average over sequential estimations. The estimator relies on packet count statistics from regular data frames. In our implementation, we consider a rate of one frame per second for the link monitoring frames. The PDR is calculated as

$PDR_{\text{WMEWMA},i} = \alpha\, PDR_{\text{WMEWMA},i-1} + (1-\alpha)\,\text{WMEWMA}_i$. As proposed by Baccour et al. in [14], we set the smoothing factor $\alpha = 0.6$ and calculate the instantaneous PDR (WMEWMA_i) over a window size of five received packets.

SNR [5]: The SNR-based estimator relies on the SNR at the receiver to estimate the PDR. This metric is shown to be more accurate than simply relying on the RSSI [1]. An a priori known correlation between the average SNR of the packet and the PDR is used to estimate the link quality. This correlation is approximated by a linear fit. Calibrated on our platform, the correlation fit is given by $PDR_{\text{SNR}} = 0.12\,\text{dSNR} - 1.7$, where dSNR is the average signal power of a packet divided by the noise floor around the packet in dB.

Four-Bit [9]: Four-Bit is closely related to WMEWMA but applies an additional filtering step. For unicast transmissions the value FourBit is calculated as $\text{FourBit}_i = \alpha\,\text{FourBit}_{i-1} + (1-\alpha)\left(\frac{1}{\text{WMEWMA}_i} - 1\right)$, and the PDR is estimated as a function of $PDR_{\text{Four-Bit},\,i} = \frac{1}{1+\text{FourBit}_i}$.

CEPS [2]: Similar to BLITZ, CEPS relies on chip errors to estimate the PDR. However, CEPS relies on the payload symbols whereas BLITZ models the errors from the preamble. As proposed in [2], the correlation between the chip errors and the PDR is approximated by a linear fit. In this case, the PDR is calculated as $PDR_{\text{CEPS}} = 1 - \frac{\text{CEPS}}{\text{Chiplimit}}$, where CEPS is the average number of chip errors per payload symbol and Chiplimit is a threshold that we calibrate to 3.44 from our measurements. For $\text{CEPS} > \text{Chiplimit}$, PDR_{CEPS} is set to zero.

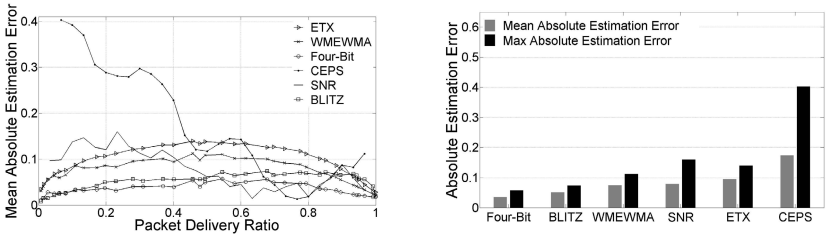
5.2 Evaluation Methodology

To ensure a systematic and reproducible evaluation of the estimators, we follow an evaluation methodology in which we transmit and record real packets over radios while the estimators are implemented in Matlab and evaluated offline by reproducing the recorded traces in Matlab. This approach has the advantage that we can replay the exact same packet traces that were recorded under identical link conditions for all estimators, making a direct comparison possible. Furthermore, this approach has more flexibility and allows determining the true PDR at a particular time from observation windows that encompass historical but also future packet events which would not be possible when determining the accuracy of the estimator in real-time.

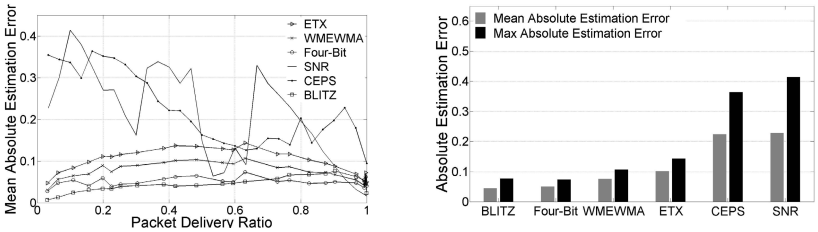
To evaluate the accuracy of an estimator, we use the absolute error. The absolute error at packet k is defined as

$$\text{absolute error}(k) = \left|\text{PDR}_{\text{true}}(k) - \text{PDR}_{\text{estimated}}(k)\right|,$$

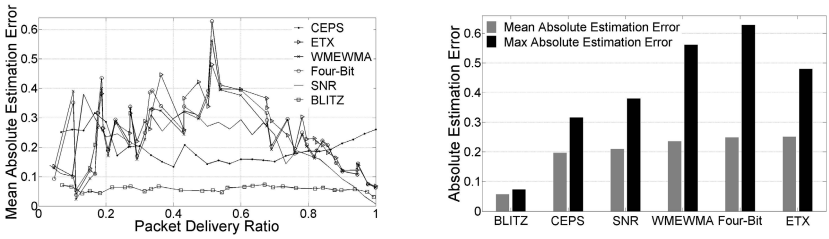
where $\text{PDR}_{\text{true}}(k)$ is the true PDR at the k-th packet and $\text{PDR}_{\text{estimated}}(k)$ is the PDR estimated by the link quality estimator at the k-th packet. The true PDR is assessed by counting the number or correctly received packets over a sliding estimation window of size w centered at packet k. The window size w is set to 100 packets for all experiments.



(a) Cable.



(b) Static.



(c) Mobile.

Fig. 6. Comparison of mean estimator error for (a) cable, (b) static, and (c) mobile links

This value provides a reasonable amount of packets to assure that the average PDR over the window has converged enough to the expected mean. Furthermore, it is assured that the window size is small enough such that the coherence time of the channel is larger than the window time, ensuring that the channel is always in steady state over the whole observation window.

5.3 Evaluation of Estimator Accuracy

Figure 6 presents the results of the estimation error comparison in the (a) cable, (b) static, and (c) mobile settings. The left figures show the mean absolute estimation error versus the true PDR. The right figures show a ranking from left (lowest error) to right (highest error) of the mean estimation error averaged over all PDR values.

The cable scenario is characterized as a very stable link condition because unpredictable and uncertain factors (e.g. small scale fading) as observed for wireless links do not occur. Therefore, the cable experiments lead to optimal link conditions for any estimator and the results can be regarded as a benchmark for the best-case performance. Not

surprisingly, the estimators that rely on packet count statistics (Four-Bit, WMEWMA, and or ETX) perform quite well in this case. Four-Bit shows the best performance as a double filtered packet statistic based estimator with an average mean absolute estimation error of 3.5 %. Remarkable is the performance of BLITZ that even shows smaller errors than packet statistic approaches like WMEWMA or ETX over cable. SNR-based estimator is expected to perform well in cable scenarios. This is confirmed by the average mean estimator error of 7.9 %, leading to the 4-th rank. Interestingly, the performance of CEPS with an average mean estimation error of 18 % is the worst. Note however the large difference in error for links with a PDR above and below 50 %. CEPS tends to be better at estimating good quality links. This result is expected considering that CEPS only assesses the chip errors in the payload of packets which tend to be rare in low quality links since most packets are lost due to synchronization failures.

Turning to the wireless static link scenario (Figure 6 (b)), we obtain a new order in the ranking. BLITZ is the winner and shows the smallest average error of 4.7 %. Since the link can still be assumed as relatively stable, the packet count statistics based approaches should perform further on well. The ranking of these estimators reflect this as the second, third and the fourth best estimators are Four-Bit, WMEWMA and ETX with average mean estimation errors below 10 %. CEPS still does not show superior results compared to packet count statistics based estimators while it now slightly beats SNR-based estimator.

Last, and most interesting are the wireless mobile scenarios. These scenarios are clearly the most challenging for all estimators as the link conditions may vary quickly and the estimators must hence be able to estimate the link quality rapidly. Since the packet count statistics estimators are the slowest, we can expect that they will perform here worst. This assumption is confirmed looking at the results of the mobile scenario in Figure 6 (c). ETX, Four-Bit, and WMEWMA are this time ranked last. Compared to the static scenarios, they have now a much higher absolute estimation errors between 23 % and 26 %. The average absolute estimation error of BLITZ remains low at 5.7 %. Despite the weighted average over the last six estimations at the cost of reactivity, BLITZ is still reactive enough to outperform all other estimators. Between BLITZ and the packet count statistics estimators are ranked CEPS and SNR-based estimator with average mean estimation errors of 19.7 % and 20.1 %, respectively. Note that the mobility experiments have a relatively low node speed of maximum 1 cm/s for the sake of comparison. We intentionally kept the node mobility low so to calculate the true PDR. We expect an even more pronounced performance gap in favor of BLITZ for settings with higher mobility.

5.4 Evaluation of Estimator Reactivity

Another important aspect of link quality estimation is reactivity. By reactivity, we refer to the ability of the estimators to quickly determine a new estimation of the PDR. An overview of the temporal reactivity for all considered estimators is given in Table 1 assuming 26 bytes packets. BLITZ is able to estimate a new PDR value after 0.13 ms, the duration of one preamble. Depending on whether the signal strength is calculated over the preamble or over the entire packet, SNR has reactivities of 0.13 or 1 ms respectively. CEPS has reactivity of 1 ms. The packet statistic based estimators provide a

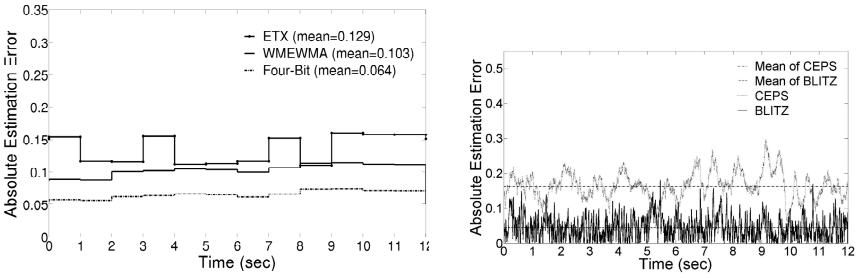
new estimation after 10 s (ETX) and $\geq$ 5 s (WMEMWA and Four-Bit). Table 1 further gives the mean error converging time of the estimators as defined as the average time it takes for the estimation error to drop below an absolute error of 0.15 for links with PDR equal to 50%.

Table 1. Reactivity of different estimators assuming 26 bytes packets

Estimator	Input	Window size	Time for first estimation	Mean error convergence time
BLITZ	chip errors	$\geq$ 1 preamble	0.13 ms	1.9 ms
ETX	packet statistics	10 packets	10 s	10 s
WMEWMA	packet statistics	$\geq$ 5 packets	5 s	6.7 s
SNR	signal strength	1 preamble or 1 packet	0.13 or 1 ms	0.13 or 1 ms
Four-Bit	hybrid	$\geq$ 5 packets	5 s	6.1 s
CEPS	chip errors	1 packet	1 ms	1 ms

5.5 Evaluation of Estimator Stability

A third important factor of the estimators is their stability. By stability, we refer to ability to provide a stable estimation of the PDR when the link conditions are static. Note that to some extent, this requirement is in contradiction to reactivity. Figure 7 illustrates the stability of the different estimators by showing the absolute error over a few seconds for a static link with true PDR of approximately 50%. While the packet statistics based estimators generally provide better stability, the stability of BLITZ and CEPS is still remarkable given their high reactivity.



(a) Packet statistic based link quality estimators (b) Chip error based link quality estimators

Fig. 7. Tracking over time of the absolute estimation error for a link with 50 % PDR for (a) ETX, WMEWMA, Four-Bit and (b) BLITZ, CEPS

5.6 Summary of the Results

To conclude the evaluation in this section, we summarize the following main findings:

– In cable scenarios, i.e., when the stable link conditions for the estimators are optimal, BLITZ is competitive to the best-performing packet count statistic estimators.

- In static wireless scenarios, BLITZ has the best performance with an average PDR estimation error of $4.7\,\%$.
- BLITZ is particularly suitable in mobile settings where it outperforms all other estimators by an absolute estimation error of $10\,\%$ to $20\,\%$ on average.
- Because BLITZ relies on chip errors in the preamble, it manages to estimate significantly more quickly and accurately the PDR compared to CEPS which estimates the PDR based on chip errors in the payload.
- BLITZ manages to estimate the PDR of low quality links accurately and within just a few milliseconds.

6 Limitations

BLITZ considers only physical layer information. This comes at some limitations. First is the issue of multiple nodes transmitting data to one receiver. In its current implementation, BLITZ does not infer which is the transmitting node on a per-frame basis for data with failed synchronization. Potential workarounds include introducing a form of physical address, using timing information, or signal fingerprints to differentiate between the different transmitters. A second issue is the interference originated by collisions. Since collisions usually occur because two or more stations start to transmit at the same time, it tends to cause further physical synchronization and chip errors. This work analyzed only scenarios without interference and further research is necessary to understand the performance of BLITZ with multiple transmitters in the same collision domain.

7 Related Work

The authors in [15] suggest that the chip error rate might be a better channel quality indicator than signal power based metrics particularly in the presence of interference. However they do not propose any estimator nor do they evaluate the feasibility to estimate the PDR from chip error measurements as we do in this work. CEPS [2] models the PDR from chip errors in the payload of successfully received packets. In contrast, BLITZ models the PDR from chip error measurements in the synchronization phase. We show in this paper that the approach of BLITZ is much more accurate and faster than CEPS because packet errors are more probable due to synchronization failures than due to chip/symbol errors in the payload.

The link quality indication (LQI) measurement [16] is a characterization of the quality of a received packet on CC2420 hardware chips. The RSSI LQI reports the received signal strength. However, to reduce the impact of narrowband interference inside the channel, the CC2420 also provides an LQI based on the average correlation value of the first 8 incoming symbols following the start frame delimiter (SFD) which presumably relates to the chip error rate as observed in the payload. Unfortunately, the LQI is a proprietary hardware implementation for the CC2420 chip and there is no exact description presented in the datasheet that could be used for comparison on other chips or on software-defined radios. According to various independent studies [17–19], the LQI on the CC2420 further suffers trustworthiness due to variations of the correlation

coefficient between LQI and PDR. Relating the LQI directly to a PDR performance therefore results in unreliable and unstable predictions.

Chen et al. propose in [20] to leverage codes to estimate the number of bit errors in the payload and model the packet delivery. The proposed technique focuses on WiFi transmissions where bit errors in the payload are common. This approach is not well suited for IEEE 802.15.4 communication because packets rarely get lost because of bit errors in the payload but rather due to synchronization failures in the preamble as we show in Section 3. Halperin et al. [5] propose a model that relies on channel state information measurements to predict 802.11 packet delivery from wireless channel measurements. This model is specific to OFDM, and not applicable to DSSS communication.

8 Conclusions

We have explored various properties of preamble symbols to design a fast, accurate and stable link quality estimator. With a software-defined radio based implementation of IEEE 802.15.4, we showed that chip errors in the preamble symbols serve as a good indicator to predict a link's PDR in point-to-point communication. Our estimator, BLITZ, is a chip error-based estimator that weights and filters sequential estimations on a per-packet level. BLITZ proved to be at least three times faster than state-of-the-art rapid estimators and more accurate than other link quality estimators under various wireless channel conditions as static, mobile, poor and good quality links. In average, BLITZ showed half the absolute estimation error (5%) of the best performing packet statistics-based link quality estimators and four times less error than signal strength and payload chip error-based estimators. Especially in dynamic environments like mobile scenarios, BLITZ is particularly superior due to its ability to react to fast changing link qualities and to exploit information from packets that fail at the frame synchronization phase.

To the best of our knowledge, there exist no commercial hardware radio chips that report the number of chip errors in the preamble symbols. The application of BLITZ is therefore currently limited to software-defined radios in which chip-level information is accessible from software. However, we do not expect any fundamental challenge in implementing such a chip error indicator on radio chips. Indicators like the RSSI or the LQI, which are similar in complexity, have for example already been integrated on the CC2420 low-power radio chips. We hope that our work will motivate chip manufacturers to implement chip error-based indicators in next-generation radio chips in order to provide higher-layer applications with a powerful metric to estimate quickly and accurately the wireless link quality.

References

1. Baccour, N., Mottola, L., Niga, Z., Boano, Alves, M.: Radio Link Quality Estimation in Wireless Sensor Networks: a Survey. ACM Transactions on Sensor Networks (2012)
2. Heinzer, P., Lenders, V., Legendre, F.: Fast and Accurate Packet Delivery Estimation based on DSSS Chip Errors. In: IEEE INFOCOM 2012, Orlando, Florida, USA (March 2012)
3. Aguayo, D., Bicket, J., Biswas, S., Judd, G., Morris, R.: Link-level Measurements from an 802.11b Mesh Network. In: Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM SIGCOMM 2004. ACM, New York (2004)

4. Reis, C., Mahajan, R., Wetherall, D., Zahorjan, J.: Measurement-based models of delivery and interference in static wireless networks. In: SIGCOMM Computer and Communications Review (2006)
5. Halperin, D., Hu, W., Sheth, A., Wetherall, D.: Predictable 802.11 packet delivery from wireless channel measurements. SIGCOMM Comput. Commun. Rev. 40 (August 2010)
6. Woo, A., Culler, D.: Evaluation of efficient link reliability estimators for low-power wireless networks. Technical Report UCB/CSD-03-1270, EECS Department, University of California, Berkeley (2003)
7. De Couto, D.S.J., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multi-hop wireless routing. In: Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom 2003), San Diego, California (September 2003)
8. Cerpa, A., Wong, J.L., Potkonjak, M., Estrin, D.: Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In: Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2005, ACM, New York (2005)
9. Fonseca, R., Gnawali, O., Jamieson, K., Levis, P.: Four-bit wireless link estimation. In: Proceedings of the Sixth Workshop on Hot Topics in Networks, HotNets VI (2007)
10. 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)
11. Boano, C., Zuniga, M., Voigt, T., Willig, A., Romer, K.: The triangle metric: Fast link quality estimation for mobile wireless sensor networks. In: 2010 Proceedings of 19th International Conference on Computer Communications and Networks, ICCCN (August 2010)
12. IEEE: IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)
13. Schmid, T.: Gnu radio 802.15. 4 en-and decoding. Technical report, UCLA NESL (2005)
14. Baccour, N., Koubaa, A., Ben Jamaa, M., Youssef, H., Zuniga, M., Alves, M.: A comparative simulation study of link quality estimators in wireless sensor networks. In: IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems, MASCOTS 2009 (September 2009)
15. Qin, Y., He, Z., Voigt, T.: Towards accurate and agile link quality estimation in wireless sensor networks. In: IEEE Med-Hoc-Net (2011)
16. Chipcon: CC2420 - 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver (2009)
17. Zhao, J., Govindan, R.: Understanding packet delivery performance in dense wireless sensor networks. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys 2003. ACM, New York (2003)
18. Ganesan, D., Krishnamachari, B., Woo, A., Culler, D., Estrin, D., Wicker, S.: Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical report, Citeseer (2002)
19. Srinivasan, K., Dutta, P., Tavakoli, A., Levis, P.: An empirical study of low-power wireless. ACM Trans. Sen. Netw. 6(2) (March 2010)
20. Chen, B., Zhou, Z., Zhao, Y., Yu, H.: Efficient error estimating coding: feasibility and applications. ACM SIGCOMM Computer Communication Review 40 (2010)

Understanding Link Dynamics in Wireless Sensor Networks with Dynamically Steerable Directional Antennas

Thiemo Voigt[1,2], Luca Mottola[2,3], and Kasun Hewage[1]

[1] Uppsala University, Sweden
[2] Swedish Institute of Computer Science (SICS), Kista, Sweden
[3] Politecnico di Milano, Italy
{thiemo,luca}@sics.se, kasun.hewage@it.uu.se

Abstract. By radiating the power in the direction of choice, electronically-switched directional (ESD) antennas can reduce network contention and avoid packet loss. There exists some ESD antennas for wireless sensor networks, but so far researchers have mainly evaluated their directionality. There are no studies regarding the link dynamics of ESD antennas, in particular not for indoor deployments and other scenarios where nodes are not necessarily in line of sight. Our long-term experiments confirm that previous findings that have demonstrated the dependence of angle-of-arrival on channel frequency also hold for directional transmissions with ESD antennas. This is important for the design of protocols for wireless sensor networks with ESD antennas: the best antenna direction, *i.e.*, the direction that leads to the highest packet reception rate and signal strength at the receiver, is not stable but varies over time and with the selected IEEE 802.15.4 channel. As this requires protocols to incorporate some form of adaptation, we present an intentionally simple and yet efficient mechanism for selecting the best antenna direction at run-time with an energy overhead below 2% compared to standard omni-directional transmissions.

1 Introduction

For many wireless sensor network (WSN) applications, reliability and energy-efficiency are among the most critical issues. Electronically-switched directional (ESD) antennas are able to steer the radiated power in specific directions that can be selected via software. By concentrating the radiated power in one direction, ESD antennas enable lower contention and extended range without additional energy consumption. Lower contention avoids packet loss and hence reduces the need for retransmissions, which saves energy. While currently ESD antennas are not frequently used in WSNs, we have demonstrated their potential by showing that only slight modifications to Contiki Collect [1], a traditional CTP-like convergecast protocol, make it efficient for ESD antennas and lead to higher packet yield at lower duty cycle [2].

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 115–130, 2013.

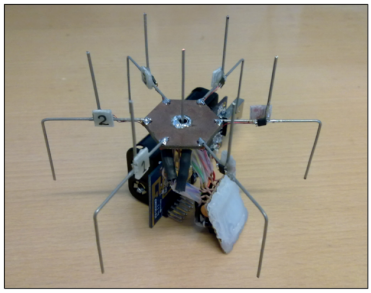
Recently, researchers have designed several ESD and other types of directional antennas for WSNs [3,4,5,6]. Most measurements with these antennas, however, have only assessed the directionality of the antenna, showing that the received signal strength of nodes that are in the direction in which the antenna radiates its maximum power is indeed higher than that of nodes at other locations. These measurements have been short-lived and were mostly conducted outdoors in open space where the receiving nodes have been placed in line of sight (LOS) of the transmitter. While these measurements fulfil the purpose of demonstrating the main property of ESD antennas, *i.e.*, directionality, they provide little insight on the link dynamics.

The goal of our study is to gain a more thorough understanding of the dynamics of link quality that is needed when developing protocols for networks with ESD antennas. In the absence of studies like the one we present in this paper, researchers made implicit assumptions about the stability of links [7,8]. Our results show that, for example, many of the existing mechanisms to select the preferred antenna configuration need to be complemented with mechanisms that continuously evaluate if the current selection of active antenna directions is still the preferred configuration, or if adaption is needed.

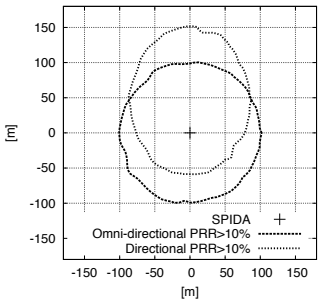
In this paper, we therefore conduct long-term indoor experiments with SPIDA, an ESD antenna designed specifically for WSNs [5]. Via a simple software API, the SPIDA antenna can be configured to steer the radiated power in six different directions, in addition to the possibility of performing traditional omni-directional transmissions. Our experiments also include nodes that are not in LOS of the transmitter.

Our experimental results show that in particular in non-LOS conditions, the best antenna direction, *i.e.*, the direction that leads to the highest packet reception rate and signal strength at the receiver, is not necessarily related to the physical position of the receiver with respect to the transmitter. It rather depends on multi-path effects. Furthermore, the best antenna direction is not stable but varies over time which requires protocols to adapt. Since multi-path induced fading varies with frequency, the antenna direction that performs best on one channel does not necessarily perform well on another channel. These results confirm that previous findings that have shown that channel frequency and changes in the environment that happen over time have an affect on the best reception angle [9] also hold for directional transmissions with ESD antennas.

Based on these results, we find that protocols for networking with ESD antennas need to adapt to the varying environmental conditions. We show that the received signal strength ($RSSI$) is a very good indicator for the success of future transmissions when using a certain antenna direction to transmit to a neighbouring node. Based on this insight, we present a simple and energy-efficient mechanism to identify the best antenna direction. Its overhead is below 2% compared to a standard omni-directional transmission with a common WSN MAC layer. We achieve this performance using a train of consecutive packets quickly transmitted within the same radio activation. This mechanism can serve as a building block for adaptive protocols for WSNs with ESD antennas.



(a) SPIDA prototype, connected to a TMote Sky node



(b) SPIDA: area where the packet reception rate is larger than 10% [2]

Fig. 1. SPIDA prototype and area of directional main lobe

Our main contributions are the following:

- We study the long-term behaviour of ESD antennas in indoor and non-LOS situations showing that the best antenna direction may change over time and that the best direction is not necessarily the same for all 802.15.4 channels as previous results suggest [9]. These results imply that protocols for networking with ESD antennas need to adapt.
- We demonstrate that an intentionally simple and efficient mechanism that selects the best antenna direction based on the *RSSI* of one packet for each antenna direction is sufficient to choose the best antenna direction with high probability.

Our findings provide valuable input for designers of protocols for sensor networks with ESD antennas.

The paper proceeds as follows. In the next section we present the SPIDA antenna. Section 3 discusses our methodology and experimental setup. The following sections discuss spatial, temporal and inter-channel variations. In Section 7 we present a mechanism to find the best performing antenna direction at runtime. Before concluding, we discuss related work in Section 8.

2 Antenna Prototype

We use a *switched parasitic element* antenna [10] called SPIDA, designed by Nilsson at SICS specifically for WSNs [5]. The antenna operates in the 2.4 GHz ISM band, matching the most common WSN radio technology. In our prototype, we integrate SPIDA with a standard TMote Sky node [11]. To be compliant with the requirements of WSNs, SPIDA has a small form factor, comparable in size to the sensor node itself, as Figure 1(a) shows.

SPIDA has one central active element surrounded by six "parasitic" elements. The central element is a traditional omni-directional antenna, implemented as a quarter-wavelength whip antenna. The parasitic elements can be switched

between ground and isolation: when grounded, they work as reflectors of radiated power; when isolated, they operate as directors of radiated power. The parasitic elements are individually controllable, yielding six possible "switches" to control the shape and direction of the antenna main lobe. They may also be all isolated: in this case, SPIDA behaves as an omni-directional antenna, which simplifies broadcasting and neighbour discovery.

The antenna gain is designed to smoothly vary as an offset circle from approximately 7 dB to −4 dB in the horizontal plane, with the highest gain in the direction of the isolated element(s). Particularly, when the n^{th} parasitic element is isolated and all other are grounded, SPIDA operates in the most directional manner, and we say that SPIDA has *direction n*. Figure 1(b) exemplifies the performance in packet reception rate (*PRR*) with this specific configuration, based on an empirical model we derived earlier [12]. In the picture, the isolated parasitic element is directed upwards. Compared to the omni-directional SPIDA configuration, the area where at least 10% of the packets are received moves upwards and becomes slightly narrower in the orthogonal direction.

The antenna is also straightforward to manufacture, and its most expensive part is the SMA connector (about $6 in single quantities). The cost, size, and radiation characteristics of SPIDA are therefore comparable with the state of the art in directional antennas for low-power wireless [3,13,14] rendering our results of general applicability.

3 Experimental Setup and Methodology

To provide protocol designers with indications on the temporal behaviour of directional antennas across different channels, we need to create challenging conditions for our experiments. To this end, we deploy a small sensor network in an office environment, as depicted in Figure 2. Compared to outdoor scenarios, our setup includes both fixed (walls) and moving (people) obstacles. Moreover, several co-existing 802.11 and 802.15.4 networks operate in the same area which creates realistic conditions of interference. The nodes are also placed to create both line of sight (LOS) and non-LOS conditions across different devices and points in time.

We deploy a node equipped with the SPIDA antenna in an office of roughly 2 m by 4 m. All other nodes in our setup are equipped with the standard TMote Sky microstrip inverted-F antenna, and passively act as probes to monitor the transmissions from the SPIDA node. Direction 1 of the SPIDA antenna points straight to node 4. The two nodes are thus always in LOS conditions. Node 4 is the only node in our setup that enjoys such conditions throughout all experiments. Node 3 and node 5 on the other hand, may be in LOS conditions depending on the presence of people in the office, who may interrupt the line of sight. This is in particular true for node 3 where the LOS is interrupted when a person sits at the desk, whereas for node 5 there is rarely a person blocking LOS. Node 2 does not enjoy LOS as there are fixed obstacles that block LOS almost completely. Node 6 and 7 are in different offices than the SPIDA node.

People sitting in these offices may further create obstacles for transmissions in addition to the walls.

To carry out the experiments, we implement small Contiki programs for both the SPIDA node and the probes. The former controls the execution of the experiments and collects the data to compute statistics. We use the IEEE 802.15.4 channel 15 as control channel, as we expected it to be the least interfered one in our setting, based on previous experiments. The processing starts by broadcasting a `START` message that informs the probe nodes about the beginning of an experiment. This message is not accounted for in the statistics. Next, the SPIDA node starts broadcasting dummy packets with different transmission power on channels 11, 15, 19 and 23. We chose these channels since we expect 11 and 15 to have low interference, channel 19 to have moderate interference, and channel 23 to be significantly interfered. The packets are sent with an inter packet transmission time of 1/8 seconds. Each train of dummy packets consists of 10 packets. We did not opt for lower inter packet transmission times to avoid successive packet loss due to link burstiness [15].

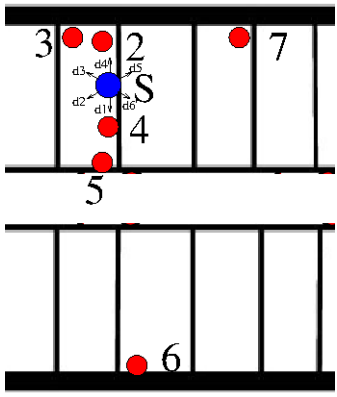


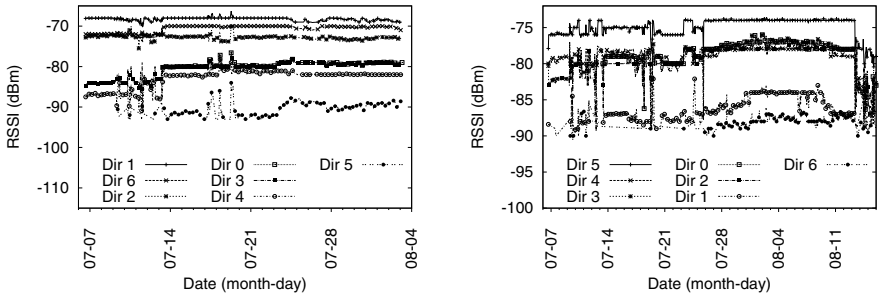
Fig. 2. Experimental setup. The node denoted with S is equipped with the SPIDA antenna. All others have microstrip inverted-F antennas, according to the standard TMote Sky design.

After receiving the 10th packet or when a timeout expires, the probes send back data on the received signal strength ($RSSI$), the link quality indicator (LQI), the noise floor and the signal to noise ratio (SNR) for each received packet to the SPIDA node that stores the data in a log file for further processing. The SPIDA node also estimates the external interference on each of the channels we use by measuring a simplified form of the channel quality (CQ) metric [16]. The CQ value is computed as the fraction of RSSI samples that are above a certain threshold. For our measurements, we have set the threshold to -77 dBm, the CC2420's default CCA threshold. The CQ value is 1 in the absence of interference. The CQ measurements confirmed our expectations on the channel quality except that it turned out that channel 15 was slightly more interfered than channel 11.

To provide a quantitative measure of the performance variations across different antenna directions, we use the statistical entropy [17] as metric. The entropy of a given random variable R is defined as:

$$H(R) = \sum_{r} -p_r * log(p_r) \tag{1}$$

Therefore, the entropy is 0 whenever a random variable always takes the same value. In our case, this means that when one direction is always performing best,



(a) Node 4: LOS conditions, TX power 5, channel 11.

(b) Node 7: non-LOS conditions, TX power 21, channel 19.

Fig. 3. Link performance along different antenna directions, compared to the omni-directional configuration. The direction of maximum gain provides an increase in *RSSI* between 10 dBm and 4 dBm.

the entropy is 0. The maximum entropy, *i.e.*, the highest degree of uncertainty on the outcome of a random variable, exists when the probability of the variable taking any value in its domain is equal.

We run experiments from early July until the middle of August. From mid July until the beginning of August, most of the offices are empty due to vacations, including those where we do not deploy nodes. Starting July 13, we also broadcast dummy packets in omni-directional mode, to obtain a baseline. Unfortunately, node 4 stopped working during the night of August 3rd, so results for it are available only partially.

4 Spatial Variations

The ability to dynamically steer the radiated power in given directions brings an additional degree of freedom to the network configuration. Protocols need to handle the possible settings depending on performance requirements. A key aspect in this regard is to understand the variability of the link performance across different antenna directions depending on the receiver's position.

Figure 3 exemplifies the *RSSI* link performance along different antenna directions, compared to the omni-directional configuration. We discuss two sample cases for LOS and non-LOS configurations and different TX power settings, although we obtained similar results in different scenarios as well.

Figure 3(a) shows the *RSSI* at node 4 when the SPIDA node is transmitting on channel 11 at TX output power 5 (around -20 dBm). The SPIDA antenna is in LOS with node 4 and the parasitic element corresponding to direction 1 points directly towards it. As expected, direction 1 indeed provides the highest *RSSI* reading at node 4 throughout the trace. The gap from the omni-directional configuration is around 10 dBm, which is significant.

Moreover, the adjacent directions 6 and 2 also yield *RSSI* values higher than the omni-directional configuration. This is a result of the SPIDA design, whose

main lobe in the direction of maximum gain is shaped as an offset circle, as shown in Figure 1(b). Accordingly, the remaining SPIDA directions result in average *RSSI* values at node 4 lower than the omni-directional configuration.

As for *PRR*, with TX power 5, node 4 receives all packets sent by the SPIDA node. Using TX power 2 (smaller than -25 dBm), node 4 receives only packets that SPIDA transmits in directions 1, 2 and 6. For direction 1, the *PRR* is almost 99% while it is 80.8% and 62.7% for directions 6 and 2, respectively. When SPIDA transmits in omnidirectional mode at TX power 2, node 4 does not receive any packets. This confirms the trends shown in Figure 3(a), but at lower TX power all SPIDA configurations but directions 1, 6, and 2, fall below the radio sensitivity.

Figure 3(b) depicts the *RSSI* performance for node 7, which is in non-LOS conditions unlike node 4. Particularly, packets transmitted to node 7 need to pass through at least two walls. The picture shows the *RSSI* readings at this node when SPIDA is transmitting on channel 19 at TX output power 21 (around -4 dBm). Over the whole trace, packets that SPIDA transmits in direction 5 are received with the highest *RSSI*: around 4 to 6 dBm higher than the omni-directional configuration. The directions 2, 3 and 4 yield a performance similar to omnidirectional. On the other hand, directions 1 and 6 shows very poor performance even though direction 6 is adjacent to the best direction 5. Most likely, multi-path effects due to non-LOS conditions make it perform worse than its counterpart direction 4 that is also adjacent to direction 5.

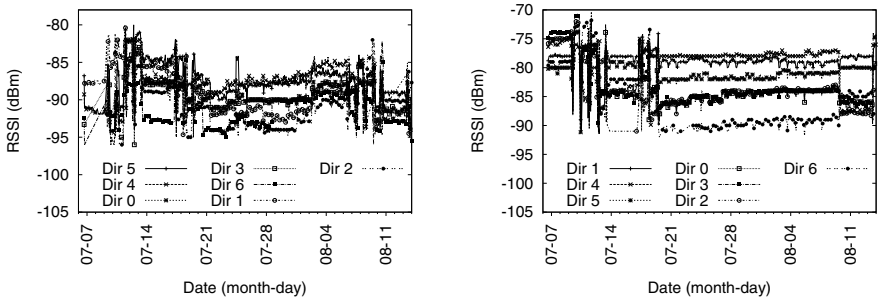
In terms of *PRR*, node 7 receives almost all packets when SPIDA transmits in direction 5. Also, directions 2, 3 and 4 have a high *PRR* of around 97.5%, while the directions with the lowest *RSSI*, direction 1 and direction 6, also have a low *PRR* of 75.3% and 51.1%. Based on the discussion above, we state the following:

Observation 1. *Using a dynamically steerable directional antenna, the best performing antenna direction is most likely related to the receiver's physical position when nodes are in LOS conditions. In LOS, side lobes also perform well. In non-LOS conditions the best direction is not necessarily related to the physical position with respect to the sender but depends on multi-path effects.*

Most protocols are unaware of this behavior, and therefore do not take advantage of it. Note that it is possible to estimate if nodes are in LOS conditions at runtime. For example, in order to estimate the distance between two sensor nodes, Pettinato et al. measure the round-trip times of ranging packets over all channels [18]. When the variance of the round-trip times across all channels is small, this is an indication that the impact of multi-path effects is low which suggests that nodes are in LOS.

5 Temporal Variations

As much as the link performance across different antenna directions shown in Figure 3 remains reasonably stable, moving obstacles and changing environmental conditions may make it vary over time. In this sense, the use of dynamically



(a) Node 6: non-LOS conditions, TX power 17, channel 19.

(b) Node 2: non-LOS conditions, TX power 5, channel 15.

Fig. 4. Link performance along different antenna directions, compared to the omni-directional configuration. The best performing antenna direction changes over time.

steerable directional antennas is not different from omni-directional ones, as we discuss in the following.

Figure 4 shows two example traces of *RSSI* readings where different antenna directions provide significantly different performance over time. We observe similar trends in different configurations as well. Figure 4(a) shows the *RSSI* at node 6 that is in non-LOS of the SPIDA, using TX power 17 on channel 19. For example, direction 4 is the best performing direction (in terms of *RSSI*) between July 20th and August 6th, but yields the worst performance in the beginning of July and some days around August 12. Direction 5 is the best direction for two smaller periods of time (July 14 to July 18 and August 11 to August 14). Moreover, direction 2 sometimes provides good performance, whereas it also results around 8 dBm worse than direction 4 for non-negligible periods of time. Direction 5 (96.7%) and direction 4 (94.1%) provide the best *PRR* over the duration of the experiment while the *PRR* of direction 2 suffers from the period of bad performance and has the lowest average *PRR* of all directions, slightly below 85%.

The figure also demonstrates periods where the performance of given antenna directions is fairly stable and other periods where there are significant instabilities, with the best antenna configuration varying distinctly over time. The times when links are unstable correspond to the when there are persons in the offices, particularly in the one where we deploy node 6.

These observations generally apply to non-LOS nodes independently of their distance from the SPIDA node. For example, Figure 4(b) shows the *RSSI* performance at node 2, using channel 15 and TX power 5 (around -20 dBm). Although the variation in the plot is not as significant as in Figure 4(a), the performance is not at all stable, especially in the early parts of the trace. Probably, the person assigned to the office where we deploy node 2 was less in his office during the duration of the experiments. While with TX power 5, the *RSSI* for most directions is not close to the sensitivity threshold at the receiver, direction 6 and direction 2 lose more than 20% resp. 10% of their packets.

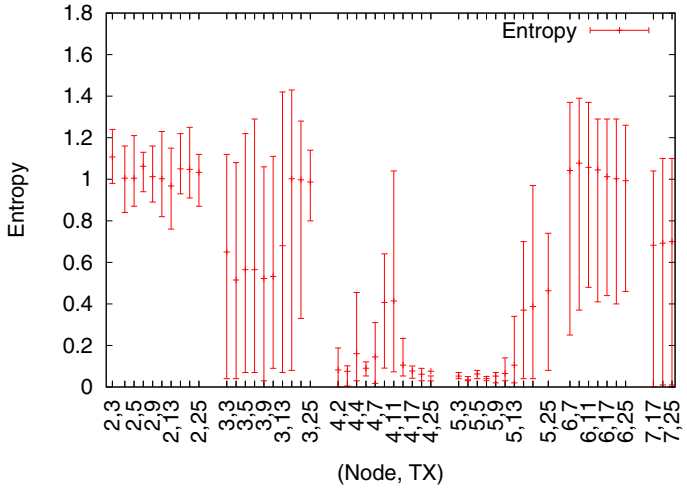


Fig. 5. Average, maximum, and minimum entropy of best performing antenna direction depending on receiver node and TX power

Based on the percentage of time a given antenna direction provides the best link performance in our experiments, we can use the entropy to cater for an aggregate indication of the temporal variations of the link performance. In our case, the maximum value of entropy would be $H = log(6) = 1.7918$. Both traces in Figure 4 thus correspond to significantly high levels of uncertainty, holding $H = 1.287$ for node 6 in Figure 4(a) and $H = 1.208$ for node 2 in Figure 4(b).

Figure 5 reports the average, maximum, and minimum entropy for all traces we collect, against the receiver node and TX power. The plot demonstrates that, depending on the receiver's position, the uncertainty in establishing the best performing antenna configuration may be significant. On the other hand, the LOS conditions again play a role: node 4 is among the ones with lowest entropy in the chart.

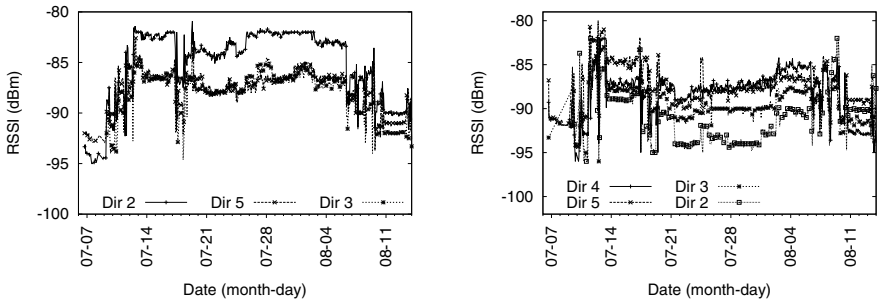
According to the findings hitherto discussed, we can claim:

Observation 2. *The link performance of* ESD *antennas is time-varying. In particular, the best antenna direction may change over time.*

This issue is, unfortunately, largely overlooked in the state of the art. Several existing protocols, for example, include a distinct discovery phase [8]. Our results demonstrate that after the initial discovery phase continuous monitoring and adaptation is necessary as the initial configuration may turn out to perform poorly in the long term.

6 Inter-channel Variations

Based on the results discussed in the previous two sections, we argue that multipath effects play a key role also depending on the specific antenna configuration.



(a) Channel 11: best performing antenna direction is 2.

(b) Channel 19: best performing antenna direction is 4 or 5.

Fig. 6. Link performance on different channels for node 6 at TX power 17. The best performing antenna direction depends on the channel.

Since different IEEE 802.15.4 channels typically imply different multi-path effects, here we study the relation between this setting in 802.15.4 networks and the link performance depending on the antenna direction.

We consider as example the link performance at node 6 using different channels and TX power 17, shown in Figure 6. Using channel 11, the best performing antenna direction is 2, as Figure 6(a) demonstrates. Direction 2 has also the highest *PRR* with 94.8% packets received, and remains the best antenna configuration most of the time. On channel 19, the situation is different: the best performing antenna directions are 4 and 5. Direction 5 also corresponds to the best *PRR* overall (96.7%), followed by direction 4 with a *PRR* of 93.4%. Using channel 19, direction 2 performs quite poorly and, in contrast to channel 11 where it performs best, never constitutes the best antenna configuration. The *PRR* using this direction is also the lowest *PRR* of all possible antenna directions (85.1%). On the contrary, using channel 11, direction 4 performs badly and achieves a *PRR* of only 75%: much lower than 93.4% on channel 19. As mentioned in Section 3, node 6 is not in the same office as the SPIDA node and hence multi-path induced fading may be severe and different for different channels.

The same discussion as above applies to other nodes. For example, at node 2, when SPIDA is configured with TX power 4, direction 5 yields a *PRR* of 99% on channel 11 while the same direction on channel 19 is the worst configuration with a *PRR* of only 73.3%. For node 7, when SPIDA's TX power is 17, direction 1 and direction 5 are the best performing directions on channel 23, both with a *PRR* of about 78%, whereas on channel 11 the *PRR* for direction 1 is only 19%.

In Figure 7 we show the average, maximum, and minimum entropy of the best performing antenna direction against receiver node and radio channel. In this case, the maximum entropy would be again $H = log(6) = 1.7918$ out of the six possible antenna directions. Two considerations emerge: *i)* varying the channel, the best performing antenna direction exhibits about the same degree of significant uncertainty as when varying the TX power; and *ii)* at some nodes, *e.g.*, node 6 and node 7, the degree of uncertainty appears to be remarkably

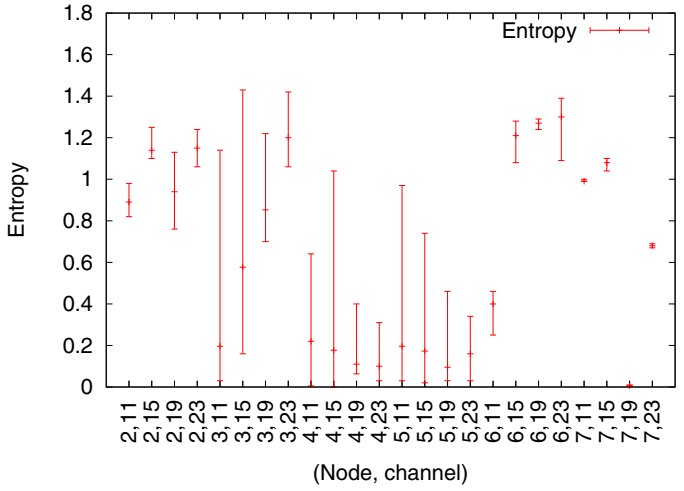


Fig. 7. Average, maximum, and minimum entropy of best performing antenna direction depending on receiver node and radio channel

consistent across different traces, which suggests that the phenomena may be independent of the specific environmental conditions during an experiment.

The discussion above leads us to the following:

Observation 3. *Due to multi-path effects that behave differently on different channels, the best performing antenna configuration on a channel does not necessarily keep the same performance on a different channel.*

This observation has implications for protocols using multiple channels. The main motivations for using multiple channels is to avoid external interference [19,20]. Our results, however, imply that when switching channels, protocols cannot retain the same antenna configuration, at least when nodes are not in LOS.

7 Taming the Added Complexity

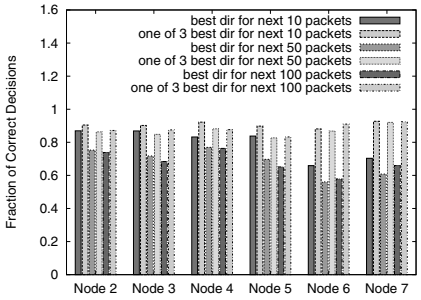
Notwithstanding the potential advantages of using dynamically steerable directional antennas in WSNs, the previous sections demonstrate that the use of this antenna technology may add to the complexity of current protocol designs. Indeed, the antenna's ability to steer the radiated power in given directions at run-time increases the degrees of freedom in configuring the network stack, as the specific antenna configuration plays a key role in determining the performance. Finding an efficient antenna configuration, on the other hand, requires adaptive mechanisms, as the best performing configurations are time varying and channel dependent.

In order to capture the characteristics of low power links, sensor network protocols use link quality estimators (LQE) [21]. The goal of such an estimator is to quickly with as little energy overhead as possible predict the probability of success of future packet transmissions. While hardware-based estimators such as the *RSSI*, *LQI* and *SNR* read values directly from the radio chip, software-based LQEs compute statistics over a number of transmissions [21]. For our purpose of differentiating between different directions, it would be ideal to use a hardware-based LQE that could identify the best direction with one packet only. In order to evaluate whether this is feasible we set up another experiment using the same nodes and positions described in Section 3. We run the experiment for a whole week from August 27 to September 3. As the experiment is conducted after the vacation period, there is in general more interference than during the experiments in the previous sections as confirmed by the lower CQ values on all channels. A higher degree of external interference makes link estimation more difficult.

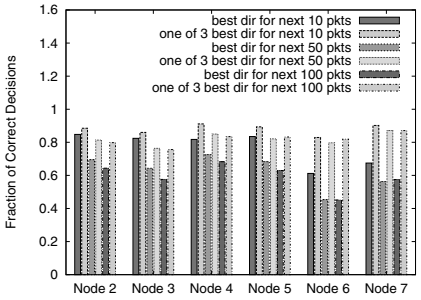
In this experiment, the SPIDA node (after the `START` message) sends 101 packets in a round-robin fashion on each antenna direction, again with an inter packet transmission time of 1/8 seconds. After the reception of the last packet or a time-out in case the last packet is lost, the probes return the *RSSI* and *LQI* of the first packet, as well as the number of packets received out of the probe messages number 2 to 11, 2 to 51 and 2 to 101. We do this over the same four channels (11, 15, 19 and 23) with selected TX output powers. The TX output powers are selected so that all nodes have some packet loss at some TX output power. We select TX output power 2, where node 4 does not receive all packets, TX output power 3, where nodes 2, 3 and 5 lose some packets and TX output power 21 where node 6 and node 7 do not receive all packets. Based on the *RSSI* or *LQI* of the first packet, we select one direction, the one with the highest *RSSI* resp. *LQI*, as the estimated best direction. Since we have the packet receptions of all directions, we can then check whether this selected direction is indeed the best one. We also check if the selected direction is among the three best directions in order to see if we have been successful in avoiding bad antenna directions.

Our results are depicted in Figure 8. The graph shows that the *RSSI* generally performs better than the *LQI* for finding the best direction. Based on the *RSSI* of only one packet in each direction, the best direction is picked between around 85% of the time for the nodes in the same office (node 2 to node 5) for the next 10 packets. For the next 50 packets, this number decreases to around 65 to 75% for these nodes and is roughly the same for the next 100 packets which indicates the stability of links. For node 6 and node 7, the results are little worse but still acceptable. Remarkable, however, is that our approach avoids picking bad directions. Figure 8(a) shows that one of the best three directions is picked with a probability of more than 90% even for the nodes in the other offices (node 6 and node 7).

We can leverage the results above to build a simple mechanism to find the best performing antenna direction at run-time, as well as to adapt to the changes in



(a) *RSSI* as direction quality estimator.



(b) *LQI* as link quality estimator.

Fig. 8. Performance of the direction quality estimators. The graphs show how often we pick the best direction or one of the three best directions for the next 10, 50 or 100 packets based on the *RSSI* or *LQI* of one packet only. Both *RSSI* and *LQI* perform well with *RSSI* performing better.

link quality that will inevitably occur. To do so, we modify the SMART strategy we proposed earlier [2].

Using SMART, every packet is quickly transmitted over all possible antenna directions, in sequence. To specify what direction is used, the transmitter appends the identifier of the active direction to the packet, encoded with a corresponding number of padding bytes. Packets can therefore be transmitted very fast, as there is no need to re-load the radio buffer: only the total packet size is changed, not the content.

At the receiver side, we measure the *RSSI* for each received packet. Once the transmitter sweeps all directions, or upon expiration of a timeout corresponding to the maximum time required to do so, the receiver transmits back an acknowledgement carrying the identifier of the antenna direction that yields the maximum *RSSI* value at the receiver.

The resulting schema is very cheap in terms of energy consumption. Compared to the average energy cost for transmitting one packet using ContikiMAC in the traditional way, our schema imposes only a 1.98% overhead [2]. Indeed, the bulk of the energy cost using state-of-the-art MAC protocols is elsewhere, *e.g.*, in the strobing phase for ContikiMAC. Because of this, our schema can be used frequently, allowing protocols to keep track of the dynamics of directional links and to execute dedicated adaptation mechanisms.

8 Related Work

In contrast to other works that investigate link layer issues for sensor networking with directional antennas, we perform long-term indoor experiments in which we study multiple channels and also include nodes that are not placed in line of sight.

We have earlier compared the performance of the SPIDA antenna against omnidirectional antennas demonstrating both that SPIDA radiates the power mostly

in the chosen directions and that the link quality is more stable [22]. We then developed a link-layer model based on empirical measurements [12]. Also Giorgetti et al. assess the improvements in link performance but with a different prototype antenna that is designed by combining four patch antennas [13]. While this study confirms our previous results none of the studies above has looked at multiple channels, temporal variations, or non LOS placement of sensor nodes.

We investigate the long term link behaviour of non-standard antennas. Others have investigated the characteristics of low power wireless links in special environments. Ceriotti et al. have studied wireless links in a jungle [23], although mostly with short-term experiments. Long-term studies of the behaviour of low-power wireless links are instead found in the literature for peculiar settings such as road tunnels [24], along with a comparison against more traditional settings.

Sensor and ad-hoc networking with directed antennas opens a lot of challenging research questions including neighbour discovery, medium access control, and routing [25]. For example, Felemban et al. present a TDMA-based MAC layer for sectored antennas that bootstraps with a very energy-consuming procedure for matching senders' and receivers' sectors [8]. Their approach has inspired us to investigate whether such a one-shot approach delivers the optimal configuration also over a longer time or whether the best configuration varies over time.

Our approach to dynamically select the best direction uses SMART that we have developed in the context of efficient data collection with directional antennas [2]. Incorporating the *RSSI* into our approach is motivated by Zuniga et al.'s study, which aims at identifying the best of a set of unreliable links [26]. They show that *RSSI* and *LQI* are better estimators than *PRR* when only a small number of packets are used for link ranking.

As noted above also ad-hoc networking researchers have used directional antennas [25]. For example, Ramanathan et al. present a fully working system [27]. Their antennas are, however, much larger and the authors do not discuss link quality issues. Navda et al. study the performance improvements possible with directional antennas for vehicular networks access [28]. They demonstrate improved throughput and longer connectivity duration compared to omni-directional antennas.

9 Conclusions

This paper has investigated the link dynamics in sensor networks with ESD antennas. We have shown that the best antenna direction, *i.e.*, the direction that leads to the highest packet reception rate and signal strength at the receiver, is not stable but varies both over time and with the selected IEEE 802.15.4 channel. Our results confirm that previous findings that have demonstrated the dependence of angle-of-arrival on channel frequency [9] also hold for directional transmissions with ESD antennas. Therefore, protocols for such networks need to be adaptive and have to be able to select the best antenna direction at run-time. Towards this end, we have presented a mechanism that performs this selection with very low overhead.

Acknowledgements. This work was carried out within WISENET (Uppsala VINN Excellence Center for Wireless Sensor Networks) funded by VINNOVA, Uppsala University, Banverket, CRL Sweden, FOI, Imego, JonDeTech, Pricer, SenseAir, SICS, TNT-Elektronik, TermoSense, VTT, Upwis, ÅAC Microtec, and WiseNet Holding. The work has been partially supported by the CONET NoE. Thanks to Martin Nilsson who designed the SPIDA antenna. and to our shepherd Tarek Abdelzaher for insightful comments.

References

1. Ko, J., Eriksson, J., Tsiftes, N., Dawson-Haggerty, S., Durvy, M., Vasseur, J.P., Terzis, A., Dunkels, A., Culler, D.: Beyond Interoperability: Pushing the Performance of Sensornet IP Stacks. In: ACM SenSys, Seattle, USA (November 2011)
2. Mottola, L., Voigt, T., Österlind, F., Picco, G.P., Quartulli, A.: Electronically-switched directional antennas for low-power wireless networks: A prototype-driven evaluation. Technical report, Swedish Institute of Computer Science (2012)
3. Choudhury, R.R., Ueda, T., Bordim, J., Vaidya, N.H.: Beamnet: An ad hoc network testbed using beamforming antennas. In: Vehicular Technology Conference (2005)
4. Felemban, E., Murawski, R., Ekici, E., Park, S., Lee, K., Park, J., Hameed, Z.: Sand: Sectored-antenna neighbor discovery protocol for wireless networks. In: SECON 2010 (June 2010)
5. Nilsson, M.: Directional antennas for wireless sensor networks. In: Scandinavian Workshop on Wireless Adhoc Networks (2009)
6. Viani, F., Lizzi, L., Donelli, M., Pregnolato, D., Oliveri, G., Massa, A.: Exploitation of parasitic smart antennas in wireless sensor networks. Technical Report DISI-11-101, University of Trento, Italy (2011)
7. Vilzmann, R., Bettstetter, C.: A survey on MAC protocols for ad hoc networks with directional antennas. In: EUNICE 2005: Networks and Applications Towards a Ubiquitously Connected World, vol. 196(1) (2006)
8. Felemban, E., Vural, S., Murawski, R., Ekici, E., Lee, K., Moon, Y., Park, S.: Samac: A cross-layer communication protocol for sensor networks with sectored antennas. IEEE Transactions on Mobile Computing 9(8), 1072–1088 (2010)
9. Zhang, Y., Brown, A.K., Malik, W.Q., Edwards, D.J.: High resolution 3-d angle of arrival determination for indoor uwb multipath propagation. IEEE Transactions on Wireless Communications 7(8), 3047–3055 (2008)
10. Thiel, D.V., Smith, S.: Switched parasitic antennas for cellular communications. Artec House, London (2002)
11. Polastre, J., Szewczyk, R., Culler, D.: Telos: Enabling ultra-low power wireless research. In: Proceedings of The Fourth International Symposium on Information Processing in Sensor Networks (IPSN 2005), Los Angeles, CA, USA (April 2005)
12. Geletu, B.S., Mottola, L., Voigt, T., Österlind, F.: Poster abstract: Modeling an electronically switchable directional antenna for low-power wireless networks. In: IPSN (2011)
13. Giorgetti, G., Cidronali, A., Gupta, S., Manes, G.: Exploiting low-cost directional antennas in 2.4 ghz ieee 802.15.4 wireless sensor networks. In: European Microwave Week (EuMW 2007), Munich, Germany, pp. 18–21 (October 2007)
14. Viani, F., Lizzi, L., Donelli, M., Pregnolato, D., Oliveri, G., Massa, A.: Exploitation of parasitic smart antennas in wireless sensor networks. Journal of Electromagnetic Waves and Applications 24(7), 993–1003 (2010)

15. Srinivasan, K., Kazandjieva, M., Agarwal, S., Levis, P.: The beta factor: Measuring wireless link burstiness. In: Proceedings of the Sixth ACM Conference on Networked Embedded Sensor Systems (SenSys 2008), Raleigh, NC, USA (November 2008)
16. Noda, C., Prabh, S., Alves, M., Boano, C., Voigt, T.: Quantifying the channel quality for interference-aware wireless sensor networks. ACM SIGBED Review 8(4), 43–48 (2011)
17. Cover, T.M., Thomas, J.A., Wiley, J., et al.: Elements of information theory, vol. 6. Wiley Online Library (1991)
18. Pettinato, P., Wirström, N., Eriksson, J., Voigt, T.: Multi-channel Two-Way Time of Flight Sensor Network Ranging. In: Picco, G.P., Heinzelman, W. (eds.) EWSN 2012. LNCS, vol. 7158, pp. 163–178. Springer, Heidelberg (2012)
19. Borms, J., Steenhaut, K., Lemmens, B.: Low-Overhead Dynamic Multi-channel MAC for Wireless Sensor Networks. In: Silva, J.S., Krishnamachari, B., Boavida, F. (eds.) EWSN 2010. LNCS, vol. 5970, pp. 81–96. Springer, Heidelberg (2010)
20. Tang, L., Sun, Y., Gurewitz, O., Johnson, D.: Em-mac: a dynamic multichannel energy-efficient mac protocol for wireless sensor networks. In: Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing (2011)
21. Baccour, N., Koubaa, A., Mottola, L., Zuinga, M., Youssef, H., Boano, C.A., Alves, M.: Radio link quality estimation in wireless sensor networks: a survey. ACM Transactions on Sensor Networks (TOSN) 8(4) (2012)
22. Öström, E., Mottola, L., Voigt, T.: Evaluation of an Electronically Switched Directional Antenna for Real-World Low-Power Wireless Networks. In: Marron, P.J., Voigt, T., Corke, P., Mottola, L. (eds.) REALWSN 2010. LNCS, vol. 6511, pp. 113–125. Springer, Heidelberg (2010)
23. Ceriotti, M., Chini, M., Murphy, A.L., Picco, G.P., Cagnacci, F., Tolhurst, B.: Motes in the Jungle: Lessons Learned from a Short-Term WSN Deployment in the Ecuador Cloud Forest. In: Marron, P.J., Voigt, T., Corke, P., Mottola, L. (eds.) REALWSN 2010. LNCS, vol. 6511, pp. 25–36. Springer, Heidelberg (2010)
24. Mottola, L., Picco, G., Ceriotti, M., Gună, Ş., Murphy, A.: Not all wireless sensor networks are created equal: A comparative study on tunnels. ACM Trans. Sen. Netw. 7(2) (September 2010)
25. Dai, H.N., Ng, K.W., Li, M., Wu, M.Y.: An overview of using directional antennas in wireless networks. International Journal of Communication Systems (November 2011)
26. Zuniga, M., Irzynska, I., Hauer, J., Voigt, T., Boano, C., Römer, K.: Link quality ranking: Getting the best out of unreliable links. In: 7th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), Barcelona, Spain (June 2011)
27. Ramanathan, R., Redi, J., Santivanez, C., Wiggins, D., Polit, S.: Ad hoc networking with directional antennas: a complete system solution. IEEE Journal on Selected Areas in Communications 23(3), 496–506 (2005)
28. Navda, V., Subramanian, A.P., Dhanasekaran, K., Timm-Giel, A., Das, S.: Mobisteer: using steerable beam directional antenna for vehicular network access. In: Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (MobiSys), San Juan, Puerto Rico (June 2007)

On Optimal Connectivity Restoration in Segmented Sensor Networks

Myounggyu Won, Radu Stoleru, Harsha Chenji, and Wei Zhang

Department of Computer Science and Engineering, Texas A&M University, USA
{mgwon,stoleru,cjh,wzhang}@cse.tamu.edu

Abstract. This paper investigates the optimal connectivity restoration in segmented sensor networks, where mobile/relay nodes are optimally placed to form "bridges" among segments, such that both the average path length from nodes to the sink and the number of mobile nodes used are minimized. We formulate the optimal connectivity restoration as a multi-objective optimization problem and develop centralized and distributed algorithms for solving it. Given global network topology information, our centralized algorithm (i.e., Cut Restoration Genetic Algorithm or CR-GA) produces a Pareto Optimal set consisting of multiple non-dominated solutions. For scenarios where the global network topology is unknown (e.g., due to unexpected network segmentation) we develop a Distributed Connectivity Restoration algorithm (i.e., DCR). DCR restores network connectivity with lower overhead (when compared with CR-GA), at the cost of a suboptimal solution (i.e., the average path length and/or number of mobiles used). Through theoretical analysis, we prove that the worst case performance of DCR is bounded. We also show the effectiveness of our solutions through extensive simulations and a proof-of-concept system implementation and evaluation.

1 Introduction

As the cost and form factor of wireless sensor nodes shrink, we envision significant growth in the demand for *enterprise-scale* wireless sensor networks (WSNs). An enterprise-scale WSN consists of disconnected subnetworks called segments, each serving its own purpose. One example application is an enterprise-scale WSN for disaster management [1], in which one sensor subnetwork identifies victims under a rubble pile, while another subnetwork monitors the stability of a damaged building. An enterprise-scale WSN may also appear in typical WSN applications. An example is a volcano monitoring application. Since it is difficult to cover the entire area of a target mountain with nodes, a plausible design option is to deploy a number of disconnected sub-sensor networks in only critical regions. To enable a system-wide analysis, *data generated in each subnetwork must be efficiently transmitted* to a remote base station. Consequently, mechanisms for optimally connecting segments are of paramount importance for enterprise-scale WSNs.

Besides segmentation in enterprise-scale WSNs because of sparse deployments, networks can often be unexpectedly segmented if many sensors become disabled.

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 131–148, 2013.

For example, unexpected network segmentation may occur when hostile users destroy sensors; when parts of the network are destroyed after a disaster, or even when environmental factors, such as wind, may arbitrarily relocate/disable sensors. It is important that the connectivity of these segmented networks must be immediately restored for correct operation.

Proactive protocols for connectivity restoration of a segmented sensor network have recently received attention [2] [3]. These protocols use more powerful nodes, called mobile/relay nodes, to build "bridges" among segments, so that the network becomes connected. These mobile nodes can be of various forms – simple wifi switches, or devices that can even fly [4]. Paying attention to the cost of mobile nodes, these schemes have focused on minimizing the number of mobile nodes. However, building bridges with the minimum number of mobile nodes may lead to suboptimal routing paths between nodes and the sink (i.e., the path length). In fact, *bridges must be carefully placed by considering several aspects of a segmented network – the sizes and shapes of segments, and even possible holes in segments.*

Two examples depicted in Figures 1(a) and 1(b) show how the geometric information of segments, and holes in segments affect the solution of connectivity restoration obtained based on the minimum number of mobile nodes, respectively. For ease of presentation, we denote static nodes by SNs, and mobile nodes by MNs hereafter. If we are to minimize the number of MNs, a single MN (denoted by triangle p) can be deployed, as shown in Figure 1(a). In this case, the average hop count for all SNs in segment s_1 to reach the sink is 9.5. However, if we connect segments s_1 and s_2 through a bridge consisting of two MNs, denoted by triangles q and r, the average hop count to reach the sink is reduced to 3.5, at the cost of one more MN. Furthermore, existing connectivity restoration schemes do not consider possible holes in a network, which may negatively influence the average hop count. Figure 1(b) illustrates an example. A connectivity restoration scheme based on the minimum number of MNs will place a single MN denoted by triangle p. A notable fact is that some packets may have to unnecessarily travel along the perimeter the hole. However, by deploying two more MNs, denoted by triangles q and r, the average hop count can be reduced (i.e., packets can now be routed over the shortcut, to reach MN p).

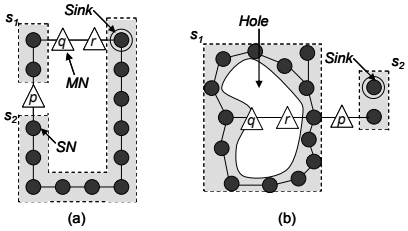


Fig. 1. The effects of (a) segment shape, and (b) holes on connectivity restoration

Additionally, protocols for connectivity restoration must be able to cope with unexpected network segmentation. More precisely, such protocols must provide mechanisms to autonomously identify network segmentation, abstract the information about segments and utilize it for optimal connectivity restoration. State-of-art protocols [2] [3] do not offer such mechanisms.

In this paper, we propose algorithms and protocols designed to address the above issues. First, we define a problem called the Optimal Connectivity Restoration

Problem (OCRP) for a segmented WSN. OCRP minimizes both the number of deployed mobiles and the average path length from nodes to the sink such that the connectivity of the segmented network is restored. This problem is formulated as a multi-objective optimization problem. Based on the observation that the problem is NP-Hard, for solving it, we propose a centralized heuristic algorithm called the Connectivity Restoration Genetic Algorithm (CR-GA). The algorithm is designed for fast convergence towards the Pareto Optimal set by using a novel scheme for efficiently generating initial solutions, fast evaluation of solution validity (based on the concept of *virtual sensor*) and a reduction of the solution search space. Furthermore, in order to handle scenarios when the global network topology, i.e., the locations of nodes and their neighbors, is not known (e.g., when a network is unexpectedly segmented) we propose a Distributed Connectivity Restoration (DCR) algorithm. DCR autonomously detects network segmentation and establishes bridges to an adjacent segment without relying on the global topology. The distributed algorithm has lower computation overhead than CR-GA, at the cost of a suboptimal solution, i.e., longer average path length from nodes to the sink and/or more mobile nodes used – through a theoretical analysis, we demonstrate that DCR has a bounded worst case performance, when compared with the globally optimal solution. Lastly, we demonstrate the efficiency and feasibility of proposed solutions through extensive simulations and a proof-of-concept system implementation, respectively.

2 Related Work

Relay node placement: The *relay node placement problem* (RNP) determines where to deploy relay nodes, RNs in short, in order to achieve various objectives. These objectives include providing *connectivity* [5] [6], *fault tolerance* [7] [8] [9], and *network lifetime* [10] [11] [12].

Lin and Xue [13] proved the hardness of the relay node placement problem *for connectivity* and proposed 5-approximation algorithm. Cheng et al. [6] proposed a faster randomized 2.5-approximation algorithm. Lloyd and Xue [5] then studied a more general problem with $R \geq r$, where R is the communication radius of RNs, and r is the communication radius of sensors. These algorithms, however, focus only on minimizing the number of relay nodes.

Some prior work pursued *fault tolerance* by ensuring that a given network is k-connected after deploying RNs [7] [8] [9]. Bredin et al. [8] presented an $\mathcal{O}(1)$-approximation algorithm for $k \geq 2$. Kashyap et al. [7] studied the fault tolerance with $k = 2$ and proposed 10-approximation algorithm. For more general case with $R \geq r$, Zhang et al. [9] proposed 14-approximation algorithm when $k = 2$.

Some researchers [10] [11] [12] focused on improving the *network lifetime* by deploying RNs. Hou et al. [10] jointly considered the energy provisioning and relay node placement with the objective of prolonging network lifetime. Wang et al. [11] studied the performance of dense WSNs when RNs are mobile. They showed that, with one mobile RN, the network lifetime can be increased by up to a factor of four. Wang et al. [12] considered the case with varying traffic, and

provided an algorithm to deploy RNs such that the network lifetime is maximized with traffic considerations. These algorithms, however, do not consider a disconnected (segmented) network.

Segmented WSNs: Abbasi et al. [14] proposed two decentralized algorithms for solving the connectivity restoration problem caused by *single node failure*. The algorithm coordinates the movement of mobile nodes in a cascading manner with the objective of minimizing the distance moved. Several work proposed to restore the connectivity of a segmented network caused by *multiple nodes' failure*. Almasaeid and Kamal [15] designed a scheme that models the movement of a mobile agent to make a segmented network connected over time. However, it is infeasible to assume that mobile nodes continuously move, because mobility consumes significant energy. Lee and Younis [2,3] considered the problem of federating disjoint segments. Especially, they focused on minimizing the number of relay nodes required to restore the connectivity. Noting that the connectivity-restoration problem is NP-hard, they provided a heuristic algorithm. Senel et al. [16] tackled the same problem by establishing a bio-inspired spider-web topology. However, these schemes focus only on minimizing the number of relay nodes.

3 System Model and Problem Formulation

We consider a disconnected wireless sensor network consisting of a set of segments denoted by $\mathcal{S} = \{s_1, s_2, ..., s_n\}$. Each segment may have holes (defined as regions without deployed nodes). In a segmented network, there are two types of deployed nodes: static nodes (SNs) denoted by the set $\mathcal{X} = \{SN_1, SN_2, ..., SN_N\}$, and mobile nodes (MNs) denoted by the set $\mathcal{Y} = \{MN_1, MN_2, ..., MN_M\}$. We assume that each node knows its location. The MNs are uniformly distributed in all segments. As we will clarify in Section 5, *we uniformly distribute MNs in segments, so that the WSN can autonomously cope with unexpected network segmentation.* Considering deployed SNs and MNs, we represent our segmented network as a HCG (Hybrid Communication Graph), formally defined as follows:

Definition 1. *A hybrid communication graph* $\mathsf{HCG}(r, \mathcal{X}, \mathcal{Y})$ *is an undirected graph with vertices* $\mathcal{X} \bigcup \mathcal{Y}$*, and edges defined as follows. Edge* e_{xy}*, where* $x, y \in \mathcal{X} \bigcup \mathcal{Y}$*, exists if and only if* $d(x, y) < \mathcal{R}$*, where* $d(x, y)$ *is the Euclidean distance between nodes* x *and* y*, and* $\mathcal{R}$ *is the communication range of a node.* ■

Each SN_i periodically senses the area of interest. Sensed data from each sensor is transmitted to the sink through the shortest path. We denote by P_i the path from SN_i to the sink and by $|P_i|$ the length of path P_i. We assume that MNs have significantly higher energy in comparison with SNs. Having defined our system model, we now formally describe the Optimal Connectivity Restoration Problem (OCRP), as follows:

Definition 2. *Given a set of SNs $\mathcal{X}$ and a set of segments $\mathcal{S}$ with holes,* OCRP *places a set of mobiles $\overline{\mathcal{Y}}$ ($\overline{\mathcal{Y}} \subseteq \mathcal{Y}$) satisfying the following three conditions: 1) $\frac{\sum_{i \in \mathcal{X}} |P_i|}{|\mathcal{X}|}$ (i.e., the average path length of all SNs) is minimized; 2) $|\overline{\mathcal{Y}}|$ is minimized; and 3) induced* HCG *is connected.* ■

In particular, the second condition of OCRP ensures the robustness against unexpected network segmentation; more specifically, by keeping more spare MNs ($i.e., \mathcal{Y} - \overline{\mathcal{Y}}$) uniformly distributed in segments, we improve the chance of autonomous network connectivity restoration (as it will be described in Section 5).

We discretize the problem by dividing the network into grid regions, where each grid is a square with side $\frac{\mathcal{R}}{2\sqrt{2}}$ ensuring that a MN in a grid can reach MNs in neighboring grids. Grids can be created by pre-computing a rectangular region that wraps a target area and dividing the rectangular region. Each node then easily determines in which grid it is located based on its location. Now the OCRP problem is to decide the grid regions where MNs will be deployed. This decision is represented by a binary variable y_{ij}, where $y_{ij} = 1$ means a MN is placed and $y_{ij} = 0$ means no MN is placed, on the grid located at (i, j). OCRP is then formulated as a multi-objective optimization problem as shown in Figure 2. The first constraint ensures network connectivity, and second and third constraints specify the ranges of variables. Finding the minimum number of MNs for restoring network connectivity is NP-Hard [2]. Hence, OCRP is NP-Hard.

$$\text{Minimize} \quad \left[\frac{\sum_{i \in \mathcal{X}} |P_i|}{|\mathcal{X}|}, \sum_{i,j} y_{ij} \right]. \tag{1}$$

$$\text{HCG is connected.} \tag{1}$$

$$y_{ij} \in \{0, 1\}. \tag{2}$$

$$\sum_{i,j} y_{ij} \leq M. \tag{3}$$

Fig. 2. OCRP problem

4 Centralized Connectivity Restoration

In this section we present a centralized algorithm, called Connectivity Restoration Genetic Algorithm (CR-GA), for solving OCRP. Given global topology information, CR-GA finds a near-optimal set of locations for MNs. Genetic algorithms are well suited for solving multi-objective optimization problems, because they can find a set of non-dominated solutions in parallel by maintaining a population of solutions [17] and they can efficiently solve NP-Hard problems [18]. Since our problem is an NP-Hard multi-objective optimization problem, we propose a genetic algorithm called CR-GA. CR-GA is designed for fast convergence to a close-to-optimal solution and uses a novel initial solution generation scheme, a virtual sensor-based solution evaluation scheme, and solution search space limitation.

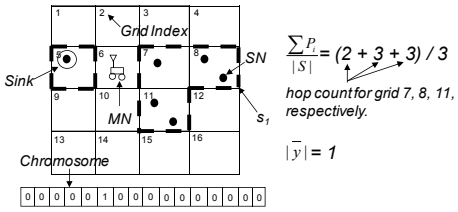


Fig. 3. A representation of a chromosome

Initial Population: Genetic algorithms represent solutions to given problems as chromosomes. A chromosome is encoded as a bit string. In our problem, each bit represents a grid in the network. A bit is set to 1 when a MN is placed in the corresponding grid; otherwise, the bit is set to 0. Given global topology information, CR-GA computes the average path length and the number of used MNs for each chromosome (i.e., chromosome's fitness or solution's optimality). However, computing the shortest paths for all SNs for each chromosome to obtain the average path length is computationally intensive. CR-GA thus uses an optional scheme for reducing the computation overhead, when nodes are relatively uniformly distributed. Consider Figure 3, which shows two segments (one containing the sink, and the other one containing five SNs) and a MN connecting the two segments. *CR-GA represents the SNs in each grid as a virtual SN at the center of the grid.* CR-GA then calculates the average path length by considering the shortest paths only for the virtual SNs in the grid network.

Having explained how the chromosome is constructed and how it's fitness is evaluated, we introduce a scheme for generating initial population of k chromosomes, where k is a system parameter. Producing *high-quality, yet diverse*, initial population is critical for fast convergence. We propose a scheme which consists of two steps. In the first step, we randomly choose a point from each segment. In the second step, we select $k_1 \in \mathbb{N}$, a parameter, and divide the 2π angle around the sink into $2\pi/k_1$ sets. Figure 4(a) shows an example with $k_1 = 4$, where different polygons represent segments. We then apply a heuristic Minimum Steiner Tree algorithm for each subregion. The first step of the scheme ensures diversity, i.e., diverse bridge locations are considered; the second step of the scheme aims to obtain high-quality initial population, i.e., the average path length from the randomly selected points to the sink are locally minimized in subregions. Figure 4(b) shows the results as a tree. We then set the bits of a chromosome corresponding to the grids intersecting with the resulting tree, if the grids are either outside segments, or inside holes in segments. We repeat the above process k times, obtaining k chromosomes – our initial population.

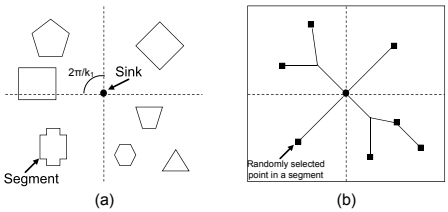


Fig. 4. (a) Initial population generation; and (b) Generated bridges

Evolution and Correction: A sequence of evolutionary processes – selection, crossover, and mutation – are applied to the initial population to produce a higher quality population. We apply the

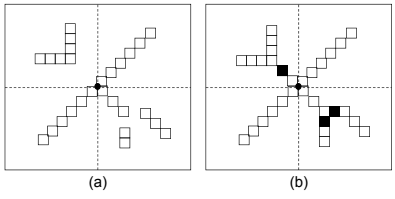


Fig. 5. Correction of a chromosome

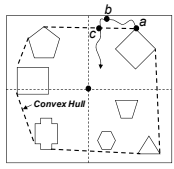


Fig. 6. Search space limitation

well-known rank-based selection algorithm [17] to implement the selection; more specifically, we rank each chromosome based on the number of dominations, e.g., if a chromosome is dominated by three chromosomes (i.e., both the number of used mobile nodes and the average path length are smaller than the three chromosomes), its rank is 3, and chromosomes with rank 0 are called the *non-dominated* chromosomes. We sort all k chromosomes in increasing rank order and select the first half. After the selection process, we randomly choose two chromosomes, say p_1 and p_2, from the selected chromosomes to perform a crossover operation. We select a position uniformly at random in a chromosome, say r. We then build a new chromosome by taking the first r bits from p_1, and the remaining bits from p_2. We repeat this operation $\frac{k}{2}$ times, creating a new set of k chromosomes. We then perform the mutation for the generated chromosomes, where we randomly select k_2 bits and switch them. After evolutionary processes are applied, some chromosomes might not satisfy our constraints. As shown in Figure 5(a), some segments are not connected to the sink. To address this problem, we first identify disconnected grids. For each such grid, we find the closest disconnected grid and connect them. Figure 5(b) shows the chromosome after the patching process. This evolution and correction process iterates until the set of non-dominated solutions converges, e.g., the algorithm stops when the set does not change for k_3 consecutive iterations.

Search Space Limitation: In order to reduce the convergence time of our algorithm, we propose to limit the search space. More precisely, we consider the placement of MNs only within the convex hull of all segments (see Figure 6 for an example) based on the theorem (we omit the proof due to space limit):

Theorem 1. *The optimal solution does not place MNs outside the convex hull of network segments.*

As described, if information about global topology is given, CR-GA obtains a set of non-dominated solutions for OCRP. However, such information may not be available, especially when a network is unexpectedly segmented due to, for example, a large number of disabled sensors by hostile users. The following section describes distributed heuristic algorithms that allow for autonomous connectivity restoration.

5 Distributed Connectivity Restoration

This section presents a distributed heuristic algorithm called the Distributed Connectivity Restoration (DCR) algorithm. The DCR algorithm establishes locally optimal bridge(s) between two adjacent segments without considering all segments in a network; thus, DCR has lower overhead (when compared with CR-GA), at the cost of a suboptimal solution (possibly longer paths from nodes to the sink and/or more MNs used), allowing any MN to compute the solution for OCRP. We first describe an overview of the algorithm.

The DCR algorithm consists of mainly three phases. In the first phase, nodes autonomously detect network segmentation and find the *boundary information* of the segment they belong to. The second phase delivers the boundary information to an adjacent segment. Since this information can not be delivered via packet transmissions because the network is segmented, our protocol uses the concept of *ferrying* – one MN in a disconnected segment stores the boundary information and moves towards the sink until it meets an adjacent segment. It is important to observe that since mobility involves very high energy consumption, it may be difficult to move all the way to the sink, especially for large scale networks. Upon reaching an adjacent segment, the ferry performs the third phase, where it finds the locally optimal (i.e., between two adjacent segments) solution for OCRP. The following sections describe the details of each phase.

Detection and Abstraction of Segments: Nodes can detect segmentation through various methods, such as distributed network cut detection algorithms [19]. Once a node detects network segmentation, it broadcasts a control packet to nodes in the disconnected segment it belongs to. Upon receiving this control packet, nodes in the disconnected segment execute a boundary detection algorithm, e.g., [20] to find the boundary nodes of the segment. When the boundary node detection phase is finished, the boundary node with the largest ID becomes the leader. This leader node stores the locations of the boundary nodes and then broadcasts its ID to the MNs in the segment.

Movement of a Ferry: Upon receiving the ID of the leader, MNs inform the leader of their remaining energy. The leader then selects a MN with the largest remaining energy and sends the locations of the boundary nodes to the selected MN. The selected MN, after receiving this information, starts the ferrying process, by traveling towards the sink until it meets an adjacent segment.

When a ferry reaches an adjacent segment, it checks *the state of the segment* – A segment's state is *disconnected* when all nodes in the segment are disconnected from the sink; otherwise, *connected*. If the state is *connected*, then the ferry executes the third phase of the DCR algorithm, which finds a locally optimal set of locations for MNs, that connects the two adjacent segments. If the state is *disconnected*, the ferry waits until the state changes to *connected*.

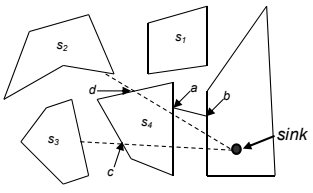


Fig. 7. An example of ferry movement in DCR

Consider Figure 7 for an example. Assume that network segmentation resulted in four segments denoted by $\{s_1, s_2, s_3, s_4\}$. Assume that s_2 first sends a ferry along the dotted line towards the sink. This ferry meets a node at point d and checks the state of segment s_4, which is *disconnected*, because it is not yet connected to the segment containing the sink. Thus, this ferry waits until the state changes to *connected*. Next, assume that segment s_4 sends a ferry.

This ferry reaches the segment containing the sink, and decides the location of bridge $\overline{ab}$. The state of segment s_4 changes to *connected*; and the waiting ferry sent from segment s_2 now builds a locally optimal bridge by running the third phase of the DCR algorithm, described in the following section.

Computation of Locally Optimal Solution: This section explains the details of the third phase, summarized in Algorithm 1. The computation of a locally optimal solution involves three major steps: 1) candidate grids selection; 2) bridge placement on holes; and 3) bridge selection.

Algorithm 1 DCR: code for ferry f

```
1: if s_c reached then
2:     // Step 1; VisEdge(s_c, s_d): Return grids on
   visible edges of s_c and s_d.
3:     {V_c, V_d} ← VisEdge(s_c, s_d).
4:     // Step 2
5:     for each g_d ∈ V_d, compute h(g_d).
6:     if s_c is connected then
7:         // Step 3
8:         for each (g_d, g_c) pair, g_d ∈ V_d, g_c ∈ V_c,
9:           compute (nm, pl).
10:        for each nm, find pl_min.
11:        for each nm, compute μ.
12:        find (g_d, g_c, h(g_d)) s.t. μ is maximized.
13:    else
14:        wait until s_2 is connected.
```

We are given two adjacent segments: one in a *disconnected* state denoted by s_d, and the other one in a *connected* state denoted by s_c. A ferry sees a network as a set of grid regions, as explained in Section 3 (See Figure 8). Define a set of grids that are contained in segments s_d and s_c by G_d and G_s, respectively. In particular, one grid in s_c is called the *destination grid* and denoted by t. The destination grid is either a grid containing the sink when the sink is in s_c, or a grid containing the entry point of a bridge that connects to other connected segment when the sink is not in s_c.

The first step of the algorithm is to find a set of *target grids* for adjacent segments s_c and s_d. Given s_d and s_c, we first find edges visible to each other. Consider Figure 8 for an example. The two segments are represented by triangles $\triangle abc$ and $\triangle def$. In this example, the visible edges are $\{\overline{bc}, \overline{ca}\}$ for segment s_c, and $\{\overline{de}\}$ for segment s_d. Target grids are the grids that are located on the visible edges of the two segments. We denote the set of target grids for s_d by V_d, and for s_c by V_c.

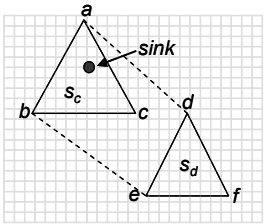


Fig. 8. An illustration of visible edges

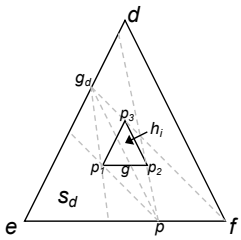


Fig. 9. Bridge placement on holes

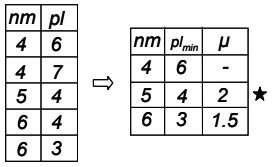


Fig. 10. An example for marginal utility computation

In the second step, the algorithm places bridges over holes in the segment. For each $g_d \in V_d$ and each hole h_i, invisible edges of hole h_i from g_d are identified. See Figure 9 for an example. By drawing two tangent lines from g_d to hole h_i, we can find that line segments $\overline{p_1p_2}$ and $\overline{p_2p_3}$ are the invisible edges. Define the set of grids on the invisible edges for hole h_i by V_{h_i}. Now for each $g \in V_{h_i}$, we consider a line starting from g_d, passing through g. We denote the farthest intersection with the edges of segment s_d by p as shown in Figure 9 (there may exist multiple such intersections). If line segment $\overline{gp}$ intersects other holes, h_i is not considered. We then consider two tangent lines from p to hole h_i. These two tangent lines, with the edges of hole h_i and possibly with the edges of segment s_d, create a region A_g, which represents the number of grids that will contribute to the reduction of the average path length by placing the bridge on that hole. For example, the two tangent lines from p (i.e., $\overrightarrow{pp_1}$ and $\overrightarrow{pp_2}$) create a region $A_g = \{p, p_1, p_2\}$. We then select g' from V_{h_i} such that A_g is maximized. We denote such grid g' for each $g_d (\in V_d)$ by $h(g_d)$.

In the third step, we consider line segment $\overline{g_dg_c}$ for each $g_d (\in V_d)$ and $g_c (\in V_c)$ as a bridge connecting two adjacent segments s_d and s_c. If line segment $\overline{g_dg_c}$ intersects any of the visible edges, the line segment is not considered. Now for each pair (g_d, g_c), representing a bridge, we compute the average path length denoted by pl and the number of used MNs denoted by nm as follows: $pl = \frac{\sum_{g \in G_d}(d(g,g_d)+d(g_d,g_c)+d(g_c,t))}{|G_d|}$, where nm represents the number of grids on $\overline{g_dg_c}$. Here the term $d(p, q)$ refers to the length of the shortest path connecting p and q. In particular, for computing $d(g, g_d)$, we consider two cases: (1) placing bridges on holes according to pre-computed $h(g_d)$ (i.e., placing MNs on line segment $\overline{g_dh(g_d)}$ that is within hole(s)); (2) not placing bridges on holes. After computing pl and nm for all (g_d, g_c) pairs, we have a set of (nm, pl) pairs (the table on the left-hand side of Figure 10 gives an example). Different from CR-GA (which produces a Pareto Frontier), due to the lack of computational capabilities, the DCR algorithm chooses one pair that maximizes the marginal utility. Marginal utility shows the incremental contribution of each added MN to the average path length. Choosing the solution with maximum marginal utility thus leads to the most economic decision. For example, for each nm, we first find the minimum pl, denoted by pl_{min}. The table on the right-hand side shows pairs (mn, pl_{min}). For each pair (mn, pl_{min}), we then compute the marginal utility, denoted by μ, as follows: $\mu = \frac{pl - pl_{min}}{nm - nm_{min}}$. In our example, from all the pairs (mn, pl_{min}), our DCR algorithm selects $(5, 4)$, the most economic decision.

6 Algorithms Analysis

As presented in Section 5, the DCR algorithm finds a locally optimal solution for *two adjacent* segments. If we consider *all* segments in a network, however,

a simple combination of locally optimal solutions may not guarantee optimal performance. Thus, in this section, we address the following research question: *how much worse is the performance of the DCR algorithm, when compared with the centralized CR-GA?*

For answering the question, we consider a network with a circular shape centered at the sink. The radius of the network is r, where $r \gg 1$. As mentioned in Section 3, there are n segments in the network. Considering a very large network (i.e., $r \gg 1$) for deriving worst-case bounds, segments and MNs are represented as points in the network.

We first identify the worst case scenario for the DCR algorithm and analyze how much worse it is, when compared with the globally optimal solution. The following lemma proves the worst-case average path length and number of used MNs for the DCR algorithm.

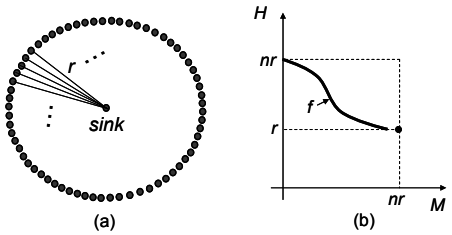


Fig. 11. (a) Worst case scenario; (b) The domain and codomain of Pareto Frontier

Lemma 1. *The DCR algorithm shows the worst performance when n segments are uniformly positioned on the circumference of the network.*

Proof. Figure 11(a) shows the worst case scenario. It is easy to note that, for this scenario, the average path length is r, and the number of used MNs is nr. Assume by contradiction that there is a scenario with either the average path length greater than r, or the number of used MNs greater than Nr. In order to have the average path length greater than r, there must be at least one segment with its path length greater than r. Since, for this scenario, the DCR algorithm places a bridge as a straight line towards the sink, the path length cannot be greater than r, i.e., a contradiction. Similarly, in order to have the number of used MNs greater than nr, we must have at least one bridge with more than r MNs; a bridge with more than r MNs is no longer a straight line. ■

We define a two dimensional Cartesian coordinate system with the domain (i.e., X-axis, and denoted by M) representing the number of used MNs and the codomain (i.e., Y-axis, and denoted by H) representing the average path length. Then, the Pareto frontier, i.e., *the solution of CR-GA*, is a curve represented by a function $f : M \rightarrow H$. We are interested in the maximum distance between any point on the curve (a CR-GA solution) and the point that represents the worst-case DCR solution (i.e., as obtained by Lemma 1). We call this distance *performance gap*. The main idea for obtaining the maximum performance gap is to bound the domain and codomain of function f. The following two lemmas find the bounds for the domain and codomain of function f, respectively.

Lemma 2. *The domain M of f is bounded by $0 < M \leq nr$.*

Proof. Since the path length from any segment to the sink for CR-GA is greater or equal to r, the average path length for CR-GA is greater or equal to r, i.e., $H \geq r$. Assume by contradiction that $M > nr$. Then, we have $M > nr$ and $H \geq r$, which means that any solution for CR-GA (i.e., points on the curve f) is worse than the solution obtained by the DCR algorithm (i.e., both the number of used MNs and average path length are greater than the DCR algorithm). ■

Lemma 3. *The codomain M of f is bounded by $r \leq H \leq nr$.*

Proof. By Lemma 2, we know that $H \geq r$. Since the domain of f is bounded by nr, the average path length is maximized when all paths from segments are aggregated into a single path of length nr. ■

Theorem 2. *The performance gap is bounded by* $nr\sqrt{1 + (\frac{n-1}{n})^2}$

Proof. Based on Lemma 2 and Lemma 3, the Pareto optimal curve f can be one of any possible curves defined in $0 < M \leq nr$ and $r < H \leq nr$, as shown in Figure 11(b). Thus, the maximum distance from point (nr, r) to curve f is the distance from point (nr, r) to point $(0, nr)$, which is $nr\sqrt{1 + (\frac{n-1}{n})^2}$. ■

Theorem 2 shows that the performance of DCR degrades asymptotically linearly with the number of segments n and the network diameter r. The interpretation of this result is that, since DCR builds bridges based on adjacent segments without taking into account all segments in the network, the overall performance degrades when there are more segments. Besides, if the diameter of a network is large, the distances between segments and the sink are more likely to be longer; thus, the performance degrades, because a better solution may be found by aggregating such long paths. However, this result also proves that the performance does not degrade arbitrarily, only linearly with the number of segments and the network diameter.

7 Simulation Results

For performance evaluation, we consider a 2,000m × 2,000m area with randomly generated segments of different sizes and shapes. Sensor nodes are uniformly deployed in each segment. To account for more realistic wireless communication, we adopt the radio model [21], which defines the *degree of irregularity* (DOI) as the maximum radio range variation in the direction of radio propagation. In our experiments, the radio range of a node is 40m with DOI=0.4, resulting in a network density of approximately 8 nodes/radio range.

We implemented DCR, CR-GA, and the state-of-art cut restoration scheme called Cell-based Optimized Relay node Placement (CORP) [2] in C++. CORP is a state-of-the-art centralized heuristic algorithm for restoring network connectivity by using the fewest MNs possible. For fair performance comparison between DCR and CR-GA, we select a CR-GA solution on the Pareto Optimal set (i.e., an average path length and the corresponding number of used

MNs) with the largest marginal utility. We used the following values for CR-GA: $k_1 = 4, k_2 = 10$ % of total bits, and $k_3 = 15$. CR-GA was executed on a PC with 64bit Ubuntu, Intel Core i7 CPU, and 8 GByte memory.

For our evaluation, we measured the average path length in hops and the number of used MNs by varying several properties related to a segmented sensor network: Segment Size (SS), Number of Segments (NS), Location of Sink (LS), and Hole Size (HS). A segment was represented as a polygon. The vertices of the polygon were selected within a randomly located circle with radius SS. SS is thus used to control the size of a segment. We ensure that the area covered by a segment is at least 20% of that of a circle with radius SS. The parameter LS represents the distance between the sink and the center of the network. The default values for our parameters were: SS=200, NS=4, LS=0, HS=0.

Evaluation of CR-GA: In each iteration of CR-GA, a set of chromosomes are generated. CR-GA computes the rank of each chromosome based on the average path lengths and numbers of MNs of the set of chromosomes. Our proposed virtual sensor (VS) can significantly reduce the time to compute chromosome's rank, when nodes are uniformly distributed. To verify this, we compared the time taken by CR-GA for computing ranks when VS is used, with the time taken when VS is not used. Figure 12 presents the results. As shown, the average rank-computation time increased linearly with the number of nodes. In contrast, CR-GA that uses VS showed a constantly small average rank computation time.

Figure 13 shows the Pareto Optimal Set obtained for a network with the default setting. Each point of the graph represents a feasible solution. As the graph shows, when we can afford a large number of MNs, we can achieve a better average path length by placing more MNs; in contrast, a solution that uses a small number of MNs has a longer average path length, but the spare MNs can be used for detecting unexpected network separation, increasing robustness.

Effect of Number of Segments: In this section we investigate how the Number of Segments (NS) affects the performance of the three protocols. We varied NS from 4 to 12, while having all other parameters set to default values. Figure 14 and Figure 15 show the average path length and the number of used MNs for the three protocols, respectively. Comparing CR-GA and CORP, we observed that both used a similar number of MNs regardless of NS. However, CR-GA produced much smaller average path length up to 20%. The reason is that, while CORP tries to minimize only the number of used MNs, CR-GA minimizes both the average path length and the number of used MNs. It should be noted that CR-GA involves higher computation than CORP, because it is based on a genetic algorithm. However, what really important is higher network performance achieved by optimizing both the number of MNs and average path length, because the computation of CR-GA is performed only once in a powerful device. Comparing CR-GA and DCR, we observe that DCR produced a slightly smaller average path length, i.e., about 4%. The reason is that DCR builds multiple bridges towards the sink without merging them. The smaller average path length, however, required a significantly higher (about 35%) number

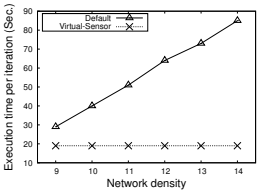


Fig. 12. Computation speed w/ and w/o VS

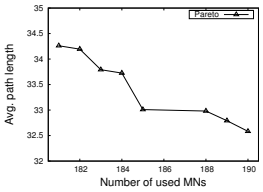


Fig. 13. Pareto Optimal set for default settings

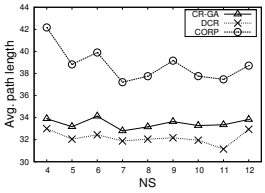


Fig. 14. Effect of NS on average path length

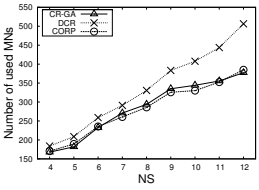


Fig. 15. Effect of NS on number of mobiles

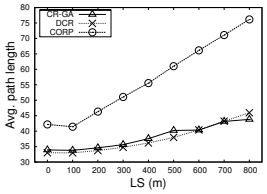


Fig. 16. Effect of LS on average path length

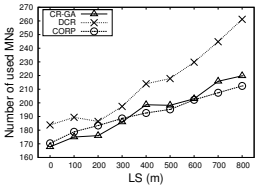


Fig. 17. Effect of LS on number of mobiles

of used MNs. An interesting observation was that the difference in the number of MNs used by the two protocols increased as NS increased. We believe this result confirms our theoretical analysis which shows that the performance gap between CR-GA and DCR increases with the number of segments.

Effect of Sink Location: In this section we investigate how the Location of the Sink (LS) affects the performance of the three protocols. We select sink locations to be LS meters away from the center of the network, towards one corner of the network. We set all other parameters to their default values. Figure 16 and Figure 17 show the average path length and the number of used MNs for the three protocols, respectively. One immediate observation was that the average path length and the number of used MNs for all three protocols increased as we increased LS. This is simply because when the sink is located far from the center of the network, a packet must travel longer distance to reach it. Comparing CR-GA and CORP, we found that the two protocols used a similar number of MNs. However, CR-GA produced much smaller average path lengths up to 75%. An interesting observation was that CORP's performance was more significantly affected by LS, than CR-GA; more precisely, while the average path length of CR-GA gradually increased with increasing LS, the average path length of CORP increased more steeply. The reason is that CORP is designed to build bridges towards the center of the network. Next, we compared CR-GA with DCR. While they achieved similar average path lengths, DCR used more MNs up to 20%. The reason is the same as above, namely that DCR builds bridges towards the sink without merging them. In fact, CR-GA finds a balance between the number of MNs and the average path length by appropriately merging bridges. An interesting observation was that the difference between the number of MNs used by DCR and CR-GA increased with increasing LS. The reason is that,

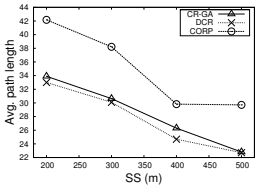


Fig. 18. Effect of SS on average path length

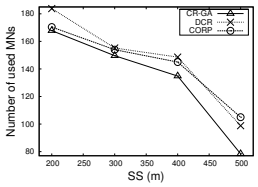


Fig. 19. Effect of SS on number of mobiles

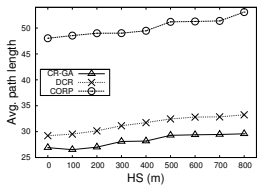


Fig. 20. Effect of HS on average path length

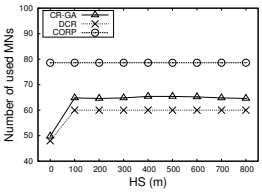


Fig. 21. Effect of HS on number of mobiles

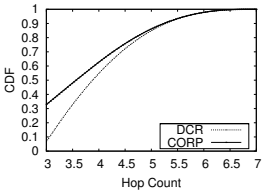


Fig. 22. CDF of hop count

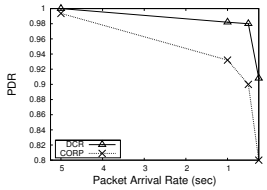


Fig. 23. Packet delivery ratio

since DCR favors building a direct bridge towards the sink, if the sink is far, it requires more MNs to connect the segment to the sink.

Effect of Segment Size: In this section we investigate the effect of Segment Size (SS). Figure 18 and Figure 19 depict the average path length and the number of used MNs for the three protocols, respectively. An immediate observation was that as we increased SS, both the average path length and the number of used MNs decreased for all protocols. The reason is simply that larger segments take more space in the network, leaving fewer empty spaces. Thus fewer MNs are required for connectivity restoration. Comparing CR-GA and DCR, we observed that both showed a similar performance in terms of the average path length. The difference comes from the number of used MNs, as CR-GA used about 10% fewer MNs on average, a relatively small improvement. This result is not surprising since the number of segments was the default value of 4. With few segments, DCR builds bridges quite well. Second, we compared CR-GA and CORP. We observed that the average path length of CORP was worse than CR-GA by up to 25%. The reason is that CORP does not consider the average path length. An interesting observation was that the difference in the number of MNs between CORP and CR-GA became larger as SS increased. We believe the reason is that CORP chooses a *representative node* (i.e., the starting point of bridges, selected for each segment) without considering the size and shape of segments in a network. Therefore, the impact of sub-optimally selected representative nodes becomes greater as the segment size becomes larger (i.e., more possible locations for selecting representative nodes).

Effect of Hole Size: To evaluate the impact of holes on protocols' performance, we consider a scenario with two large rectangular-shaped segments (200m × 1,000m), and place holes of varying sizes in them. In each segment we create a bar-shaped hole, of varying heights (i.e., bars with different sizes of

100m × [0, 800]m). Figure 20 and Figure 21 depict the average path length and the number of used MNs for the three protocols, respectively. As shown in Figure 20, for all three protocols, the average path length increased with an increasing hole size. The reason is that larger holes result in longer, detoured routing paths. It should be noted that, although DCR and CR-GA place bridges on holes, depending on the locations of bridges, there are still nodes that use detoured paths, thereby showing small increases. Comparing CR-GA and DCR, we observed that CR-GA produced about 9% smaller path length by using more MNs. The reason is that, while DCR places only a single, straight-line bridge over a hole, CR-GA places multiple bridges, or even merge bridges. Comparing CR-GA and CORP, we observed that although CR-GA places MNs on a hole, CORP used more MNs for restoring the connectivity. The reason is the large sizes of the two segments used in this experiment. As we mentioned previously, CORP more likely to choose representative nodes far from the sink, when the size of a segment is large. This suboptimal selection of representative nodes results in a large number of MNs. Furthermore, since CORP does not handle holes, it has longer average path length.

8 System Evaluation

As a proof-of-concept system, we implemented our DCR algorithm in TinyOS 2.1.1 for the TelosB platform and compared it with CORP. We deployed 10 TelosB motes in each of two segments in a disaster training facility of approximately 150m by 150m, as shown in Figure 24. Routing paths from motes to the sink were obtained using CTP [22]. Motes reported events, with varying reporting rates, i.e., 250msec, 500msec, 1sec, and 5sec, by sending a packet. We manually placed TelosB motes on the computed bridges based on DCR and CORP, using TelosB motes as MNs. As Figure 24 shows, the solid line represents the bridges for CORP, and the dotted line the bridges for DCR; filled circles represent MNs. We measured the path length in hops and computed the packet delivery ratio at the sink.

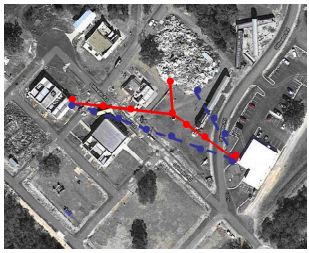


Fig. 24. A deployment area at a Disaster Training Facility

Figure 22 shows the cumulative distribution function for path length (in hops) for DCR and CORP. As shown, DCR has a smaller hop count. The reason is that CORP, to reduce the number of MNs, merged two bridges, resulting in longer paths. We then compared the Packet Delivery Ratio (PDR) of DCR and CORP. Figure 23 shows the results. For both protocols, the PDR decreased as the packet arrival interval decreased. A notable observation was that the PDR of CORP more rapidly decreased. We believe that the reason is because the merged paths increased the chance of collisions and possible congestion.

9 Conclusions

This paper investigates optimal connectivity restoration in segmented WSNs, an important, largely unexplored problem. Given the global network topology, our centralized algorithm is used to restore the network connectivity such that both the average path length and number of used mobile nodes are minimized. A distributed scheme is also developed for enabling nodes to autonomously cope with unexpected network segmentation at reduced computational costs.

Acknowledgements. We thank Dr. Marco Zuniga for shepherding this paper. This work was funded in part by NSF awards 1127449, 1145858, and 0923203.

References

1. George, S.M., Zhou, W., Chenji, H., Won, M., Lee, Y., Pazarloglou, A., Stoleru, R., Barooah, P.: DistressNet: a wireless AdHoc and sensor network architecture for situation management in disaster response. IEEE Communications (2010)
2. Lee, S., Younis, M.: Optimized relay placement to federate segments in wireless sensor networks. IEEE JSAC (2010)
3. Lee, S., Younis, M.: Recovery from multiple simultaneous failures in wireless sensor networks using minimum steiner tree. JPDC (2010)
4. Pruhit, A., Sun, Z., Mokaya, F., Zhang, P.: Sensorfly: Controlled-mobile sensing platform for indoor emergency response applications. In: IPSN (2011)
5. Lloyd, E., Xue, G.: Relay node placement in wireless sensor networks. IEEE TOC (2007)
6. Cheng, X., Du, D.-Z., Wang, L., Xu, B.: Relay sensor placement in wireless sensor networks. WINET (2008)
7. Kashyap, A., Khuller, S., Shayman, M.: Relay placement for higher order connectivity in wireless sensor networks. In: INFOCOM (2006)
8. Bredin, J.L., Demaine, E.D., Hajiaghayi, M., Rus, D.: Deploying sensor networks with guaranteed capacity and fault tolerance. In: MobiHoc (2005)
9. Zhang, S.W., Xue, G., Misra: Fault-tolerant relay node placement in wireless sensor networks: Problems and algorithms. In: INFOCOM (2007)
10. Hou, Y., Shi, Y., Sherali, H., Midkiff, S.: Prolonging sensor network lifetime with energy provisioning and relay node placement. In: SECON (2005)
11. Wang, W., Srinivasan, V., Chua, K.-C.: Extending the lifetime of wireless sensor networks through mobile relays. IEEE/ACM ToN (2008)
12. Wang, F., Wang, D., Liu, J.: Traffic-aware relay node deployment: Maximizing lifetime for data collection wireless sensor networks. IEEE TPDS (2011)
13. Lin, G.-H., Xue, G.: Steiner tree problem with minimum number of steiner points and bounded edge-length. Inf. Process. Lett. (1999)
14. Abbasi, A., Younis, M., Akkaya, K.: Movement-assisted connectivity restoration in wireless sensor and actor networks. IEEE TPDS (2009)
15. Almasaeid, H.M., Kamal, A.E.: Data delivery in fragmented wireless sensor networks using mobile agents. In: MSWiM (2007)
16. Senel, F., Younis, M., Akkaya, K.: Bio-inspired relay node placement heuristics for repairing damaged wireless sensor networks. IEEE TVT (2011)

17. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation discussion and generalization. In: Proceedings of the 5th International Conference on Genetic Algorithms (1993)
18. Kim, H., Shin, K.: Asymmetry-aware real-time distributed joint resource allocation in ieee 802.22 wrans. In: INFOCOM (2010)
19. Won, M., George, M., Stoleru, R.: Towards robustness and energy efficiency of cut detection in wireless sensor networks. Elsevier Ad Hoc Networks (2011)
20. Li, F., Luo, J., Zhang, C., Xin, S., He, Y.: Unfold: Uniform fast on-line boundary detection for dynamic 3d wireless sensor networks. In: MobiHoc (2011)
21. He, T., Huang, C., Blum, B.M., Stankovic, J.A., Abdelzaher, T.: Range-free localization schemes for large scale sensor networks. In: MobiCom (2003)
22. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection tree protocol. In: SenSys (2009)

Online Device-Level Energy Accounting for Wireless Sensor Nodes

André Sieber and Jörg Nolte

Distributed Systems/Operating Systems Group
Brandenburg University of Technology, Cottbus, Germany
{as,jon}@informatik.tu-cottbus.de

Abstract. Energy is the crucial factor for the lifetime of wireless sensor networks. Nonlinear battery effects and nonuniform workload distribution can lead to early node failures. This makes it necessary to manage energy consumption. But to manage energy it is essential to know how much energy is spent by the system. Additionally, for a more fine-grained management it is necessary, to know where the energy is spent. This can be a complicated task, since nodes are not identical due to device variations and the consumption can change over time.

In this paper we present an online energy accounting approach which focuses on simplicity instead on fine granularity and timing accuracy. We argue that the efficacy of an energy accounting model depends more on the input consumption data than on exact timing, especially when the real consumption varies between nodes and in time. Results show that this approach is capable of correctly accounting the energy that nodes spend in scenarios with deviating environment conditions.

Keywords: Energy, accounting, measurements, operating systems, embedded systems, wireless sensor networks.

1 Introduction

Energy is a major concern in wireless sensor networks. Sensor nodes should run for years with a limited energy budget providing a high quality of service, mostly using conventional alkaline or lithium batteries. To prevent sensor nodes from early failures due to depleted batteries, it is necessary to know the energy consumption and manage the available energy.

While the consumption can be predicted with simulators, e.g. [1], running tests on real hardware is more precise and flexible. Since it can not be assumed that all sensor network developers have access to expensive measurement equipment, a software based approach is feasible and flexible.

Within the development phase, the assumptions about the energy consumption of a node make it possible to determine its reachable lifetime. Since sensor network applications are based on modular operating systems like TinyOS [2], Contiki [3] or Reflex [4], the application developer uses the system and its drivers for the platform he/she is using. The developer has no deeper knowledge

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 149–164, 2013.

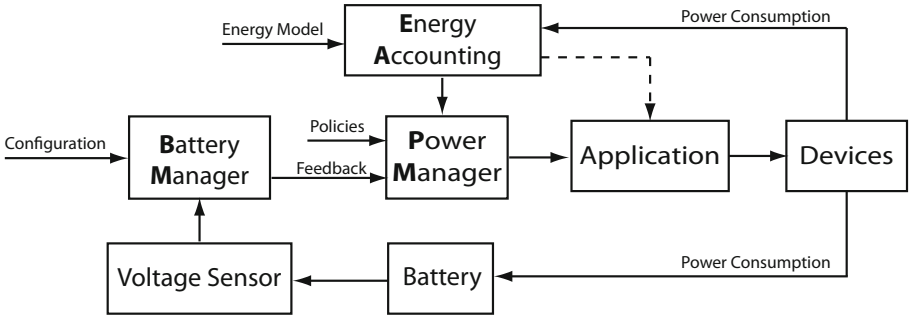


Fig. 1. Power management concept taking the current consumption and the battery state of charge into account

about the node hardware and inner behavior of the OS or the device drivers, which can lead to misleading assumptions about the sensor nodes energy consumption. Providing information about the consumption of individual devices helps the developer to find potential *energy holes* and thus increase the lifetime.

Within the actual deployment phase, online energy accounting is an instrument for the nodes power management to monitor the consumption and enforce its constraints and policies. Fig. 1 shows an approach where the consumed energy as well as information from the battery are used to control the application behavior to reach a predefined lifetime goal. Taking the battery state of charge into account makes it possible to react to the batteries non-linear effects [5] and possible inaccuracies of the consumption model. While the consumption over a certain time is interesting for energy management, the momentary consumption is interesting for the battery management which can limit the load to the battery and increase the lifetime [6].

The energy that is consumed by a sensor node depends on the devices that are active, the time these devices are active and the consumption of these devices. Which device is active is most often under control of the node software, along with the time it is active. Statements about the consumption of a device can be taken from the manufacture's datasheet. But not all nodes are equal. Besides variations of the analog and digital components, manufacturing faults can occur. While this may not render the nodes useless, it could change their power consumption footprint.

To power a node an energy source is necessary to which the node is connected directly or using a voltage regulator. Connecting the node directly exposes them to the degrading voltage level of the source, which can change the consumption foot print of the device due to voltage dependencies in the power consumption and performance of integrated circuits [7]. Additionally, the voltage can fall below the operating limit of certain devices or the entire node. Voltage regulators can avoid effects of hazardous voltage drops of batteries by delivering a constant voltage to the node and seem to make energy accounting simpler. But these regulators suffer from low efficiency [8], especially when the system drains very

little energy, e.g. when in sleep mode. Furthermore, this efficiency depends on the input voltage and the power drained through the connected consumer and must be considered in an online-consumption accounting. Together with the hardware costs, the benefits may be negated and the usage must be carefully considered by the system designer [7].

An energy accounting approach must deal with such deviations. One way to face the device variations is to generate the model for each node individually. But this is expensive and error prone. Another way is to assume and use a maximum consumption for all nodes. While this much simpler, it is way too inaccurate since outliers would rule the assumption. For most nodes the real consumption would be much lower than assumed. The third way is to incorporate measurements of the nodes and usage of a statistical model to cover most but not all nodes. Facing the variation makes it necessary to implement a dynamic accounting which is capable of changing the underlying consumption values depending on the voltage level or converter efficiency.

The remainder of the paper is structured as follows: In section 2 related work is presented. The accounting approach and how its input values were collected is presented in section 3. Implementation details are shown in section 4. In section 5 our approach is evaluated. Finally, a conclusion is given in section 6.

2 Related Work

Approaches for energy accounting can be divided into hardware based, software based and combined approaches.

The authors of [9] present a power monitoring infrastructure. While this approach is hardware based and not intended for in-field deployments, they base the need for power monitoring on device variations along with software changes that can change the power consumption as a side effect.

Coloumb counters, e.g [10], can be used to track the consumed energy online, but like other hardware based approaches they introduce an energy overhead and increase the nodes hardware cost. An in situ power observation tool based on a shunt resistor called SPOT is presented in [11]. Through iCount [12], the additional hardware costs are greatly reduced for systems featuring a voltage converter. The approach counts the switching cycles of the regulator and correlates them to the energy consumed by each switch.

Quanto [13] eliminates the disadvantage of other hardware based approaches to only measure the consumption of the whole system instead of single devices and states. Together with the information about the system consumption and the active devices within an interval, the approach uses linear regression to identify individual consumptions.

In [14] a software based approach for online energy accounting is presented. To keep track of the consumed energy, so called energy containers are introduced and refined in [15]. The approach targets the accounting of tinyDB queries. The necessary accounting infrastructure consists of finite state machines for each individual device where the states are the consuming states of the devices and the

edges represent state changes. Both include information about their consumption and duration.

Another software approach is presented in [16]. For each device activation and deactivation a timestamp is taken and the difference accumulated. The management keeps track of the devices and uses them to estimate the system energy consumption.

The authors of [1] present AEON, an off-line tool for prediction and profiling the energy consumption of Mica2 nodes. To build their energy model the current consumption of three nodes was measured, resulting in a variation of approximately 5% between them but no further details where provided.

The approach presented in [17] introduces the so called passive voltage scaling. Instead of using a DC-DC converter, the node is connected directly to the battery and the approach scales the μC frequency according to the available voltage level. Through not using a converter or statically use the μC frequency available at the lowest voltage level, the approach benefits in runtime and throughput.

In [18] the impact of device variations regarding sleep modes is analyzed. For the tested μC, an Atmel SAM3U based on an ARM Cortex M3 core, a deviation of more than factor 5 between different nodes for the sleep consumption was observed. For active modes the deviation was around 10%. The effect increased with the temperature. To overcome this, the authors proposed a variability-aware duty cycle, were the node watches its temperature and adjusts its duty cycle based on a stored sleep-power vs. temperature ratio created for each node.

While the basic idea and mechanisms of our energy accounting are similar to [14] and [16], we focus on the model input instead of trying to maximize the timing accuracy. We argue that a complex model is still vulnerable to variations. A more simple model with a pessimistic enough input can cover most variations. Additionally, our approach is capable of dealing with variations generated by the energy source.

3 Device Level Energy Accounting

When designing an energy accounting infrastructure for embedded systems, the first question is how the energy is consumed. Followed by the question how that behavior can be best modeled regarding accuracy and effort. The efficiency of the model not only depends on the time but also on the consumption values fed to the model.

3.1 Device Energy Consumption

A sensor node consists of various devices with partly independent control flows. While some devices are only binary and simply activated or deactivated, other devices have multiple states. The state changes could happen under the control of the driver or happen independently as given by the device state machine. The device activation can occur as discrete and fixed events or be time depended. Devices can be context driven and thus under the control of the application or

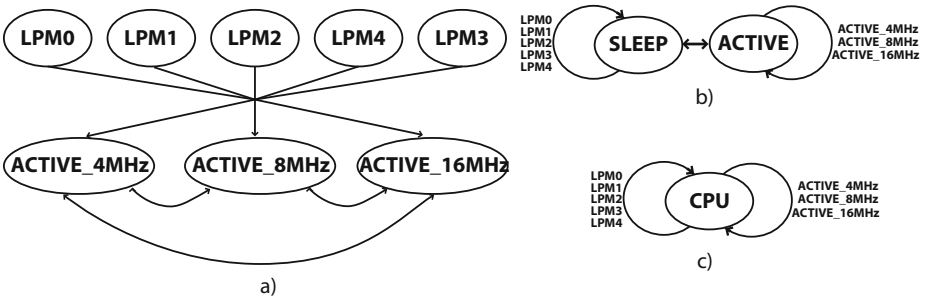


Fig. 2. Possible accounting granularity for the μC a) every sleep mode and frequency of the CPU accounted individually b) no distinction between the single sleep states and frequencies c) account all states to a single sink

demand driven when requests occur. For example, receiving over an interface is context driven while sending is demand driven. Additionally, there are different factors that can influence the device consumption. The device can have different configurations, e.g. the transmission power of the radio. The consumption can be more or less dependent on the supply voltage and/or the temperature. Some devices or device operation states have a consumption of a few μA while others consume a multiple of 10mA.

Other devices could influence the usage of a device. For example, interrupts or other cpu activity can prevent a device driver from deactivating a device and thus unexpectedly increase the device active time and, as a result, the consumption.

3.2 Accounting Model

For accounting, existing driver code has to be modified. This requires understanding of the drivers functionality. For new driver code understanding is already necessary, making the overhead for the driver developer minimal.

Our approach, like e.g. [16] is based on tracking the active time of device states.

This makes it possible to model and cover a nodes energy consumption in a simple and flexible way. To account the consumed energy of a device, the active time and its consumption must be provided. For this we adopt Quanto's *energy sinks* and *power states* view to our needs. In our view a sink is a potentially independent unit that consumes energy, while a power state defines how much energy is consumed by a sink.

The energy sinks of the individual devices form a finite state machine describing the devices energy behavior. Since the accounting granularity depends on the number of sinks in our approach, it can be reasonable to switch sinks to states and vice versa. Fig. 2 shows an example of the possible accounting granularity. For development and testing purposes a detailed coverage of the energy distribution within the CPU may be useful to find *energy holes* due to wrong

sleeping behavior. Within a productive real-world environment, the accounting can be much simpler since such a detailed breakdown may not be necessary for the power manager or network maintenance. When memory is very constrained it is possible to account all energy states to a single sink, providing information about the energy consumption of the device but loosing all information about its internal distribution. To support such flexibility, the proposed accounting system has to be able to switch between sinks and modify the consumption of a sink dynamically.

Another reason for modifying the consumption of a sink dynamically are changes of the supply voltage due to battery voltage decrease (or increase when connected to a harvesting system) when the node is connected directly. We propose a system that monitors the voltage and informs all involved device drivers when the voltage changes substantially. The driver developer is responsible for providing the content for a function that changes the device sink's consumption. The frequency in which the voltage level is checked may be changed depending on the load, since high loads can lead to voltage drops and thus changing the consumption of active devices. Using information about the active devices, the frequency can be increased as the load rises and decreased when little load is applied to conserve energy.

The accounted amount may be modified according to an efficiency factor induced by a voltage regulator. If a regulator is present, the supply voltage of the node is stable, but the efficiency of the regulator depends on its input voltage and the load applied. The factor must be recomputed when the regulator input (most likely a battery) voltage, or the load changes. The factor has to be applied to all accounting computations

3.3 Gathering Consumption Data

To build our energy model we ran a series of experiments where the different active modes of 90 Texas Instruments eZ430-Chronos [19] nodes where measured. While not explicitly designed as wireless sensor node, they seem ideal due to their features, size and low price compared to designated sensor nodes. They are equipped with an MSP CC430F6137 [20] which includes an embedded C1101 radio module and additionally features a pressure gauge and an accelerometer.To gain access to the ports we modified the nodes by swapping the buttons with pin connectors.

Setup. The measurement setup consisted of a PowerScale Unit [21] capable of measuring current dynamically in a range from 200nA to 500mA at a maximum of 100k samples/sec. To determine the accuracy of the measurement, the standard deviation σ of each mode for every node was calculated. Due to the design of the measurement hardware, σ rises with increased consumption. To reduce the impact of the temperature on the energy consumption the measurements where made in a room equipped with an air-conditioning system, keeping the temperature fixed at $25 \pm 1°C$.

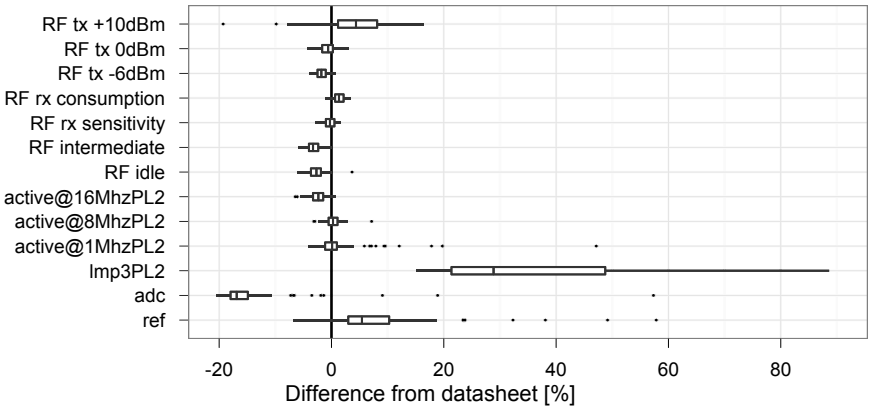


Fig. 3. Relative difference from datasheet for selected modes at 3V

The measured modes of operation include different on-chip parts of the µC. Sleep and active modes, different radio states, the consumption of the on-chip Analog-Digital-Converter (ADC) and voltage reference generator (REF) were measured. The µC features different power levels (PL0 to PL3) which determine the maximal possible frequency but depend on the voltage provided. Since PL2 is sufficient for all frequencies below 20Mhz and is enough to power all on-chip devices, it is used for building the energy accounting models. The radio modes along with the ADC and REF include the current of the µC sleep mode at PL2, which is specified by the datasheet to have a current consumption of 2.2µA at 3V.

Measurement Results. Fig. 3 shows the measured current consumption of the nodes compared to the datasheet values.

Since the measurement conditions of the datasheet values could not fully be reproduced due to the connected external components, a higher consumption can be expected and is natural.

This holds especially true for the µC sleep mode, b ut the difference varies much among the nodes in this operation mode. 20% of the sleeping nodes could not reasonably be shown in Fig. 3 since they consume a multiple of the datasheet prediction. For some of the nodes the consumption while sleeping fluctuates compared to the other nodes. This fluctuation and the high consumption indicates that these nodes are defective, but not unusable. Using them in a productive - long running and thus energy conserving - sensor network can lead to unexpected node failure due to exhausted batteries.

For all other modes the discrepancy between the measured nodes and the datasheet values is considerably lower. With rising consumption, most modes show less variance and a minor number of outliers. Depending on the operation mode, some or all nodes have a lower consumption than expected, for example most radio modes consume slightly less energy. One notable exception is the

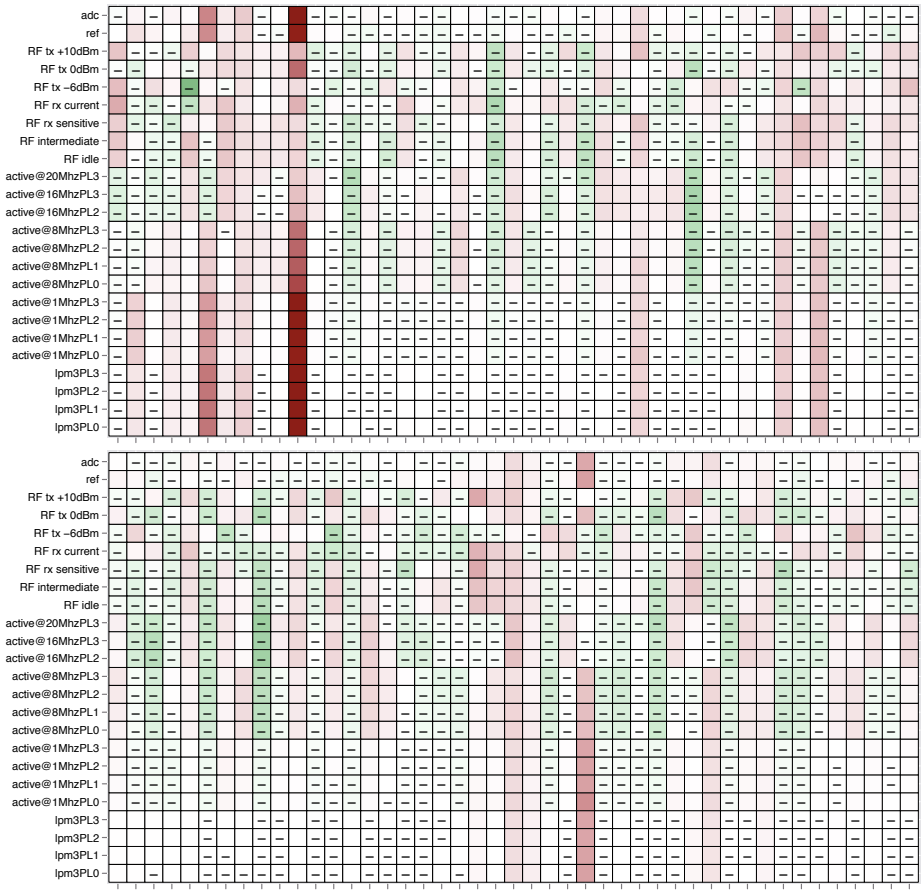


Fig. 4. Average current of each node (horizontal) in a mode (vertical). Color intensity marks the distance to the median of all nodes. "-" indicates that the mode is better than the median.

radio transmission mode with +10dB. Some of the nodes behave noticeably different than the others resulting in a wide variance. As we investigated, we identified the antenna as the problem, since some nodes have a loose antenna, resulting in a much lower consumption than expected. The nodes antenna is a metal surrounding the display, on nodes where this antenna was replaced by a simple wire based antenna, the lower consumption was also observed together with a noticeably increased range. Please note that these modified nodes are not part of the results presented and are not included in the energy model.

Differences between modules could be leveled depending on the node. For example, the REF module consumes more energy than expected on most nodes, but the ADC consumes less energy, but both modules work mostly in conjunction.

Table 1. Current consumption values at 3.0V used to build the models

	Average current in mA		
	m-Datasheet	m-Median	m-Pessimistic
LPM3 sleep	0.0022	0.0028	0.0044
CPU@8Mhz	1.75	1.755	1.770
CPU@16Mhz	3.45	3.369	3.407
ADC	0.15	0.124	0.127
REF	0.1	0.105	0.109
Radio IDLE	1.7	1.657	1.672
Radio RX	16.0	16.214	16.371
Radio TX@0dBm	16.9	17.694	17.894
Radio TX@10dBm	33.0	34.441	35.952

Since the current consumption varies, the question arises if it is likely that nodes are "bad" or "good" in all modes of operation. Fig. 4 shows the average current consumption of each node (horizontal) in every mode (vertical) compared to that modes median. The color intensity marks the distance to the modes median (darker means more deviation), while the "-" indicates that the mode is better than the median of all nodes. As the figure shows, there are nodes that are "bad" or "good", but mostly the nodes vary.

Model Building. Table 1 shows the values used to build the energy accounting models. Apart from the datasheet values used for the *m-Datasheet* model, the median of the experiment result set is used as a second model base *m-Median*. Additionally, the 0.8 percentile of the experiment result set is used to build a pessimistic model (*m-Pessimistic*). While a pessimistic model ensures that the consumption is not underestimated which is more dangerous for power management than overestimating, using the worst nodes consumption would be much too pessimistic since it consumes a multiple of the others. The 0.8 percentile covers most of the nodes while not including a huge overestimation of better nodes consumption.

As the table shows, the differences of the modes vary between the models. Mostly the models based on the experiments differ only in a few µA. The highest impact can be expected from the sleep mode values, since it takes a substantial part of the consumption for nodes with a low duty cycle.

Since the consumption accounting is, for the sake of efficiency, not based on floating point numbers, the smallest unit for the current is one µA and thus all values are rounded up. Rounding up results in an overestimation of the sleep mode but as argued, a small overestimation is not as dangerous as an underestimation may be. Additionally, using integer numbers makes it possible to use the hardware multiplier available on many µC which reduces the computation costs.

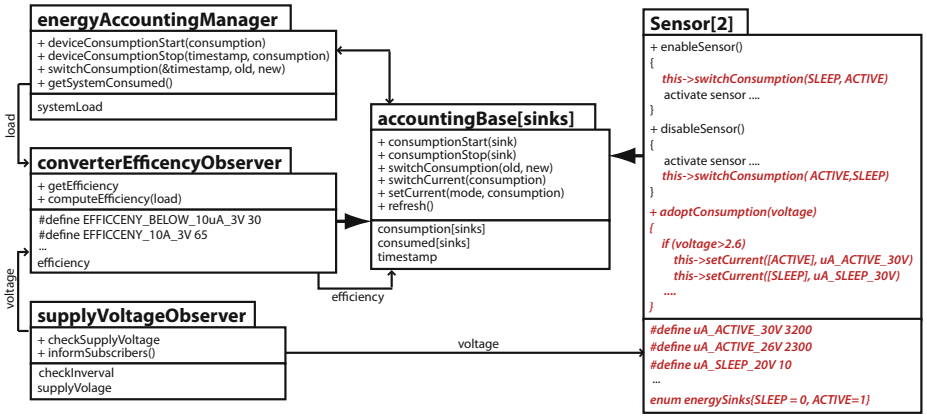


Fig. 5. Accounting system overview. The SENSOR shows which code must be added to existing driver code.

4 Implementation

While we implemented our approach for the event-driven operating system REFLEX, it can be easily adopted to other operating systems targeting embedded systems and sensor nodes. Since REFLEX is implemented in C++, it opens some opportunities to ease the integration of the accounting mechanisms into the existing device drivers. Each driver that should be accounted is derived from a base class and gains all the functions and variables necessary for the accounting. Furthermore, the device is registered at the accounting manager (*energyAccountingManager*) and, if included, by the mechanisms for the adaption of changing voltages (*supplyVoltageObserver*) and the presence of a voltage converter (*converterEfficencyObserver*).

The developer of the device driver must insert the function calls into the code parts that change the energy consumption of the device. Additionally, when the supply voltage varies and has an impact on the consumption, the developer must provide a function which changes the consumption parameters according to the input voltage.

All consumption information is stored locally at each device driver. This makes, together with the subscriber based registration of the devices by the manager and observers, the implementation flexible and changes in the granularity affect only the corresponding driver.

Fig. 5 shows the relevant parts and interfaces of the accounting system design. An example of the necessary changes to the existing driver code is also shown. The *energyAccountingManager* keeps track of the current system load. This information in needed by the *converterEfficencyObserver* to calculate the efficiency factor which is used in the calculation of the consumed energy. To track the converters base costs and adopt to the changed efficiency due to changes in

the supply voltage, the *converterEfficencyObserver* is derived from the base class as other drivers.

The accounting system was inserted into the device drivers for the Texas Instruments eZ430-Chronos nodes. As timer base for the timestamp generation we used the 32kHz timer of the MSP430. This set the resolution for time at 1/32ms. Since nodes should run for a long period of time, the timestamps needed to use 64-bit values. The consumption of the different devices and modes spread over several orders of magnitude. To cover this we used 16-bit datatypes, resulting in a minimal current consumption at 1μA to a maximal current consumption of 65.53mA. The consumed energy is stored in 64-bit values to cover the entire lifetime of a sensor node.

5 Evaluation

To evaluate the approach several experiments using typical sensor node applications were performed. If not noted otherwise, each application was executed with the three accounting models and then the actual consumption of each application is measured on two different nodes.

The first application, *app1*, let the node sleep for a long period of time. Every two minutes the node woke up to sample a sensor (the internal voltage) using the ADC. The results and the accounting data were then sent to a base station. Additionally, the node activated the radio receive mode for 50ms during each interval.

The second application, *app2*, used a low power listening scheme to sample the radio channel every 250ms. Every 10 seconds the node sent the accounting data to the base station. Additionally, every 5 seconds the supply voltage was checked.

To test the supply voltage adaption the third application *app3* was used. For this application the node was equipped with an LED which was toggled every 500ms. The node checked the supply voltage every second and sent the accounting data every 10 seconds. *app3* was only evaluated with one node and the *m-Median* model, since no statistical data about the LEDs power consumption was available and the current consumption of the used LED was measured. The *m-Median* model was compared to the actual consumption for 3.0V and 2.2V. Additionally, an experiment was performed where the voltage was decreased from 3.0 to 2.2V in steps of 0.2V.

Application 4, *app4*, utilized the radio module and the CPU active state to test a first implementation of the converter efficiency adaption. The node stayed in the 8Mhz active state for 500ms and afterwards listened for messages for another 100ms. Additionally, the accounting data was sent with the maximum transmission power. An ON Semiconductor NCP1400 [23] was used for this experiment. Fig. 6 shows its measured efficiency at various supply voltage levels. For the experiment a supply voltage of 1.5V was used. The measured efficiency was translated to 6 steps (<100μA, <1mA, <3mA, <10mA, <20mA and >20mA) for the online adaption.

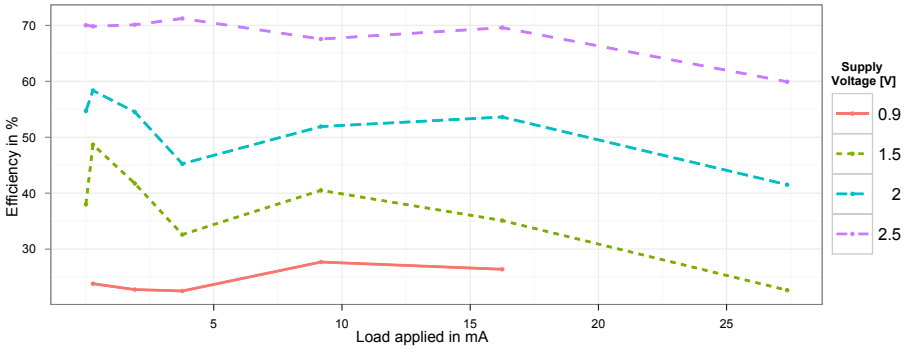


Fig. 6. Measured efficiency of step-up voltage converter NCP1400A33T1

To show how the accounting can be used by the application, *app5* monitores the consumed energy during its active phase. This information is used to dynamically calculate the sampling frequency based on available energy and desired runtime. With each invokation the node activated the radio receive mode for 50ms, sent a message to a base station and lit up an LED for 200ms.

In Table 2 the results for *app1* are shown. From the accounted consumption and runtime, the average consumption was calculated. The consumption predicted by the models is almost identical, except for the *m-Pessimistic*. It predicts a 20% higher consumption due to its parameters.

In Fig. 7 the predicted consumption is described. In all models the radio receive mode represents the biggest part of the consumption, followed by the low power sleep mode. The pessimistic model accounts for a bigger part to the sleep modes than the others. This behavior confirms to the expectations, since *app1* does little except sleeping. In comparison with the actual consumption measured for two devices, the *m-Pessimistic* model is better suited to the increased consumption of node B. Most likely this is due to a higher sleep consumption.

The results of application *app2* are shown in Table 2. As the results show, the difference between the three models is small. *m-Datasheet* and *m-Median* differ from each other by around 2%, while the difference between *m-Datasheet* and *m-Pessimistic* is 3%.

Compared with the actual measurements the *m-Datasheet* model performs quite well. It overestimates the actual consumption of node A by 2% and of node B by 1%. With 5% for node A, the *m-Pessimistic* model has the

Table 2. Average current of *app1* and *app2* compared with the accounted consumption

	Average current in mA				
	m-Datasheet	*m-Median*	*m-Pessimistic*	A	B
app1	0.01002	0.01014	0.01222	0.0098	0.0115
app2	0.32869	0.33556	0.33841	0.3221	0.3251

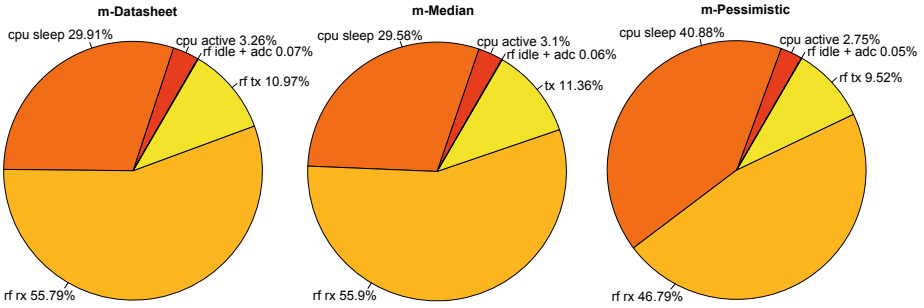


Fig. 7. Contribution of each device to the total consumption for each model with *app1*

highest overestimation. In Fig. 8 the predicted consumption is further described. The biggest part in the consumption is accounted for the radio receiving mode. Around 92% of the energy is spend to maintain the listening scheme of *app2*. The variation between the models is low, the ratio between the different consumers is almost the same. The pessimistic model accounts a little more for the sleep mode than the others, which reflects in the ratio.

As we further investigated the actual consumption, we noticed that radio receive mode behaves than expected. It seems that the actual transition to the receive state within the radio module takes almost 1ms in which the consumption is lower than the radio idle state, resulting in an overestimation of the radio consumption. We reduced this overestimation by waiting 500ns before starting the accounting for the receive mode. Additionally, sometimes and nondeterministic, the radio does not go into the receive mode after being commanded to, although the internal radio state machine thinks it is. We are currently searching for a solution to these problems to further improve the accounting mechanism.

Table 3 shows the results of the measurements of *app3*. For all three evaluated voltage scenarios, the difference between the accounted consumption and the

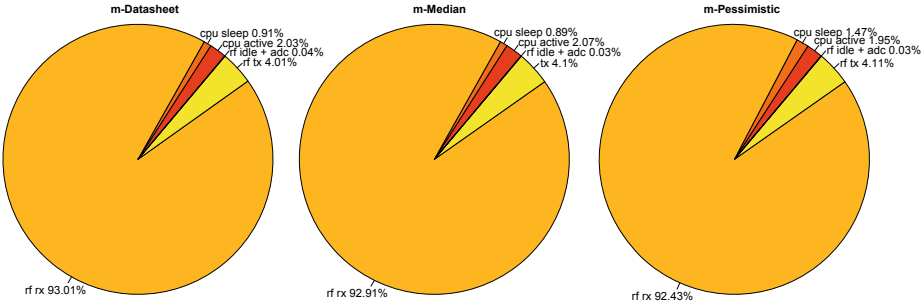


Fig. 8. Contribution of each device to the total consumption for each model with *app2*

Table 3. Current consumption of *app3* compared with the accounted consumption under different voltages using node A

	Average current in mA	
	m-Median	measured
3.0V	1.809	1.804
2.2V	0.750	0.743
3V down to 2.2V	1.280	1.272

actual measured consumption is below 1%. This is mainly motivated due to knowledge of the exact LED consumption and the huge influence of the LED on the total consumption, but shows that the *voltageObserver* works as intended.

Table 4. Current consumption of *app4* compared with the accounted consumption under different efficiency models

Average current in mA for node A			
measured	efficiency adoption	fixed worst efficiency	fixed best efficiency
5.139	5.301	8.707	4.198

In Table 4 the results of application *app4* are shown. The base consumption of the voltage converter is included in the average current. The difference between the actual consumption and the online accounting model with converter efficiency adaption is 3%. This value is considerably better than assuming a fixed efficiency. Using the best measured efficiency for 1.5V supply voltage, the consumption is underestimated by 22%. When using the worst efficiency the average current is assumed to be 68% higher than measured. While this experiment is simple and should be expanded in the future, the results show that an adaption to the converter efficiency is better suited to model the consumption than using a static model.

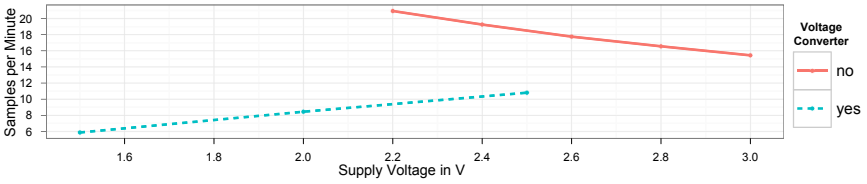


Fig. 9. Adaptive calculated sampling rate unter different voltages for *app2*

Fig. 9 shows how *app5* sets its frequency under differed supply voltages with and without a voltage converter. Without a converter the frequency increases with lower voltages since the LED consumption reduces. Although using a voltage converter makes it possible to use lower voltages, application frequency is

lower in the beginning and decreases with lower voltages due to the reduced efficiency of the converter and the stable supply voltage to the LED.

6 Conclusion and Future Work

In this paper we have shown how the online accounting of energy can be made more robust against device variations. A high number of devices were measured and the results were used to build more realistic models and thus cover more nodes by introducing only little overestimation. Our approach is designed to be flexible to fit the level of accounting to the differed needs within the development phase and the real-world deployment. In contrast to other software based accounting approaches, our system is able to adopt to the changing consumption of a device. We proposed a mechanism for taking consumption changes due to decreased voltage into account. Additionally, an approach to overcome the variable efficiency of voltage converters due to dependencies on load applied and supply voltage was proposed. This makes it possible to use accounting not only for development but also for dynamic online power management in real-world deployments.

In the future we want to expand our accounting model to other devices and nodes, especially designated sensor nodes. Furthermore, we plan to use the information about the consumption the accounting gathers for fine-grained online energy management. Additionally, the approach will be evaluated in a real sensor network project for development as well as energy monitoring and management in the deployment.

References

1. Landsiedel, O., Wehrle, K., Götz, S.: Accurate prediction of power consumption in sensor networks. In: EmNets 2005 Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors (2005)
2. Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., Culler, D.: TinyOS: An operating system for sensor networks. In: Ambient Intelligence (2004)
3. Dunkels, A., Gronvall, B., Voigt, T.: Contiki - a lightweight and fexible operating system for tiny networked sensors. In: Proceedings of the First IEEE Workshop on Embedded Networked Sensors (2004)
4. Walther, K., Nolte, J.: A Flexible Scheduling Framework for Deeply Embedded Systems. In: Proc. of 4th IEEE International Symposium on Embedded Computing (2007)
5. Linden, D.: Handbook of batteries, 2nd edn. McGraw-Hill Companies (1995)
6. Sieber, A., Nolte, J.: Device Management for Limiting the Load Applied to Batteries, 11. GI/ITG KuVS Fachgesprch Sensornetze (2012)
7. Park, C., Lahiri, K., Raghunathan, A.: Batterydischarge characteristics of wireless sensor nodes: An experimental analysis. In: Proceedings of the IEEE Conf. on Sensor and Ad-hoc Communications and Networks (SECON), Santa Clara, pp. 430–440 (2005)

8. Park, S., Saviddes, A., Srivastava, M.B.: Battery Capacity Measurement and Analysis Using Lithium Coin Cell Battery. In: Proc. Int. Symp. Low Power Electronics & Design (2001)
9. Woehrle, M., Beutel, J., Lim, R., Yuecel, M., Thiele, L.: Power monitoring and testing in wireless sensor network development. In: Workshop on Energy in Wireless Sensor Networks (2008)
10. Texas Instruments, bq26231 Low Cost Battery Coulomb Counter For Embedded Portable Applications, Webpage http://www.ti.com
11. Jiang, X., Dutta, P., Culler, D., Stoica, I.: Micro power meter for energy monitoring of wireless sensor networks at scale. In: Proceedings of the 6th International Conference on Information Processing in Sensor Networks (2007)
12. Dutta, P., Feldmeier, M., Paradiso, J., Culler, D.: Energy Metering for Free: Augmenting Switching Regulators for Real-Time Monitoring. In: Proceedings of the 7th International Conference on Information Processing in Sensor Networks (2008)
13. Fonseca, R., Dutta, P., Levis, P., Stoica, I.: Quanto: tracking energy in networked embedded systems. In: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation (2008)
14. Kellner, S., Bellosa, F.: Energy accounting support in TinyOS. 2. GI/ITG KuVS Fachgesprch Systemsoftware und Energiebewusste Systeme (2007)
15. Kellner, S.: Flexible Online Energy Accounting in TinyOS. In: Marron, P.J., Voigt, T., Corke, P., Mottola, L. (eds.) REALWSN 2010. LNCS, vol. 6511, pp. 62–73. Springer, Heidelberg (2010)
16. Dunkels, A., Osterlind, F., Tsiftes, N., He, Z.: Software-based on-line energy estimation for sensor nodes. In: Proceedings of the 4th Workshop on Embedded Networked Sensors (2007)
17. Cho, Y., Kim, Y., Chang, N.: PVS: passive voltage scaling for wireless sensor networks. In: Proceedings of the Symposium on Low Power Electronics and Design (2007)
18. Wanner, L., Apte, C., Balani, R., Gupta, P., Srivastava, M.: A case for opportunistic embedded sensing in presence of hardware power variability. In: Proceedings of the 2010 International Conference on Power Aware Computing and Systems (2010)
19. Texas Instruments, eZ430-Chronos Development Tool Datasheet, Webpage http://www.ti.com
20. Texas Instruments, CC430F61xx 16-Bit Ultra-Low-Power MCU Datasheet, Webpage http://www.ti.com
21. Hitex Development Tools GmbH, PowerScale Datasheet, Webpage http://www.hitex.com
22. HAMEG Instruments GmbH, HM8143 Datasheet, Webpage http://www.hameg.com
23. ON Semiconductor, NCP1400A Datasheet, Webpage http://onsemi.com

Coexistence Aware Clear Channel Assessment

From Theory to Practice on an FPGA SDR Platform

Peter De Valck, Lieven Tytgat, Ingrid Moerman, and Piet Demeester

Ghent University – IBCN – iMinds, Gaston Crommenlaan 8, 9050 Ghent, Belgium
peter.devalck@intec.ugent.be,
{firstname.lastname}@intec.ugent.be

Abstract. Wireless sensor networks are used by an ever growing number of applications which have ever increasing Quality of Service requirements. The available unlicensed industrial scientific and medical bands – where wireless sensor networks typically operate – are crowded with a number of technologies interfering with each other. Delivering a sufficiently high QoS within these frequency bands is therefore becoming more and more difficult. A theoretic concept named Coexistence Aware Clear Channel Assessment (CACCA) promises more reliable QoS when different technologies utilize the same. Within this paper we propose two methods to perform CACCA and create an SDR prototype to show that CACCA can achieve a high packet error rate reduction in an IEEE 802.15.4 network when it coexists with IEEE 802.11.

Keywords: Coexistence, Sensor Network, Wireless, Interference Avoidance, IEEE 802.11, IEEE 802.15.4, CCA, SDR, WARP, FPGA.

1 Introduction

The exponential growth in wireless devices during the last decade has presented a new problem for wireless sensor networks. The increased number of wireless technologies as well as the higher requirements for wireless communication put a high strain on the limited unlicensed spectrum available to these devices. Being resource and energy limited, wireless sensor nodes often get the short end of the stick when confronted with other technologies, resulting in severely degraded communication capabilities as shown in several publications [1 - 4].

Specifications in most communication standards ensure that users of the same technology are capable of coexisting in the same frequency band. Between different technologies however, this coexistence is often limited or nonexistent. Different technologies might still be needed in identical environments to support diverse needs of different applications. Eg. A wireless sensor network using the IEEE 802.15.4 technology might be co-located with IEEE 802.11 stations. The first one is capable of supporting very long term battery powered operation, while the latter is capable of delivering higher bandwidth connectivity, be it at higher energy cost. Taking a look at the 2.4 GHz band and the interaction between some common digital wireless technologies we see that some standards include support for coexistence (like Adaptive

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 165–178, 2013.

Frequency Hopping for IEEE 802.15.1 – 2005 [5]), while no such provisions exist for the IEEE 802.11 [6] or 802.15.4 [7] standards. Indeed, several studies [1 - 4] have shown that severe throughput degradation can be observed when IEEE 802.11 and IEEE 802.15.4 devices interfere.

In [2] we introduce the concept of Coexistence Aware Clear Channel Assessment (CACCA), which is capable of reducing this interference, fostering cross-technology coexistence. CACCA can be applied on a single wireless technology or on multiple interfering technologies. In the case of IEEE 802.11 and IEEE 802.15.4 devices, the paper concludes that the highest reduction in packet error rate (PER) can be achieved by deploying the CACCA mechanism on IEEE 802.11 devices. Therefore this paper will focus on the implementation of CACCA on an IEEE 802.11 device and the effects on IEEE 802.15.4 communication.

While the concept of CACCA is fully explained in the paper, several factors need to be considered before an actual implementation can be achieved. In this paper we implement CACCA as an extension to existing IEEE 802.11 systems while remaining standards compliant. We used the WARP [8] software defined radio (SDR) as our implementation platform. Keeping in mind the standard compliance and the relative simplicity of the proposed extensions it should be relatively easy to port this effort to existing or future IEEE 802.11 devices. In section II the requirements of this backwards compatible CACCA are considered and two possible implementations are proposed.

In section III a simulation of these solutions is discussed, while the actual implementation is handled in section IV. We experimentally analyze and verify both implementations in section V. Section VI mentions future research and implementation possibilities, ending with a conclusion in part VII. In the scope of this paper, we will refer to the IEEE 802.15.4 standard as ZigBee and to the IEEE 802.11g standard as Wi-Fi.

2 Requirements and Solutions

2.1 Coexistence Aware Clear Channel Assessment

In [2] the concept of Coexistence Aware Clear Channel Assessment is introduced as a means to improve coexistence between technologies. It comes down to complementing the existing CCA mechanisms of a technology with additional CCA modules capable of detecting other technologies.

In figure 1.A traditional sensing CCA operation is visualized: before transmitting, the radio will stay in receive mode for a short while and try to determine whether another user is using the channel. Depending on the used protocol, transmissions can be postponed (time based interference avoidance, figure 1.B) or moved to another channel (frequency based avoidance, figure 1.C). To sense the occupancy of the channel, both ZigBee and Wi-Fi support two CCA mechanisms, energy based CCA and preamble detection. For the first method the channel energy is compared to a predetermined threshold while the second method detects the presence of a technology specific sequence.

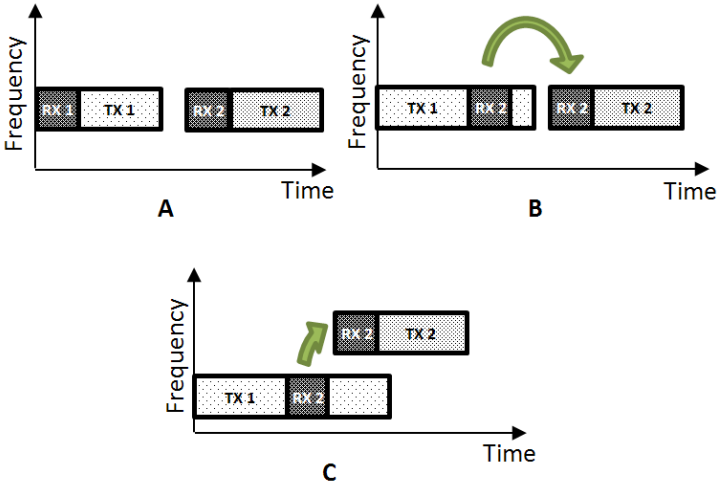


Fig. 1. Basic CCA operating principle (A) with time based avoidance (B) or frequency based avoidance (C)

Without any modification these standard CCA methods will not allow ZigBee and Wi-Fi devices to coexist. Due to the technology specific nature of preamble detection this technique is inherently incapable of detecting other technologies. The simplicity of the energy based CCA means that it could possibly detect other technologies, but in the case of Wi-Fi and ZigBee the different bandwidths and transmission levels mean that ZigBee will be overly sensitive to Wi-Fi, while Wi-Fi will be less sensitive to ZigBee. Theoretical analysis and experiments [1] confirm that both technologies indeed suffer severe throughput degradation when interfering – up to 80% depending on the exact configuration – so this energy based CCA is clearly not sufficient. Therefore a new technique is required to improve coexistence.

CACCA proposes extending a standard CCA with additional methods that are capable of detecting other technologies. This allows the sender to avoid colliding with packets from other technologies as shown in figure 2. Focusing on a ZigBee - Wi-Fi scenario, [2] concludes that the biggest throughput gains can be achieved by adding a ZigBee CACCA to existing Wi-Fi devices. Additional gains can be achieved by modifying the ZigBee CCA but this was left as a future improvement. This paper will mainly focus on adding a ZigBee friendly CACCA to a Wi-Fi device.

As mentioned in the introduction, additional design decisions had to be taken to get from the conceptual CACCA presented in [2] to a working implementation. In figure 3 a modified Wi-Fi system is outlined: in a traditional Wi-Fi setup the received signal is passed through an aliasing filter and digitized, after which CCA is performed. Thanks to the high bandwidth of a Wi-Fi receiver, the same digital signal can be used for the detection of ZigBee signals. As each Wi-Fi channel covers multiple ZigBee channels, CCA must be performed on all contained ZigBee channels. To achieve this, the high bandwidth Wi-Fi channel is mixed down to several low bandwidth ZigBee

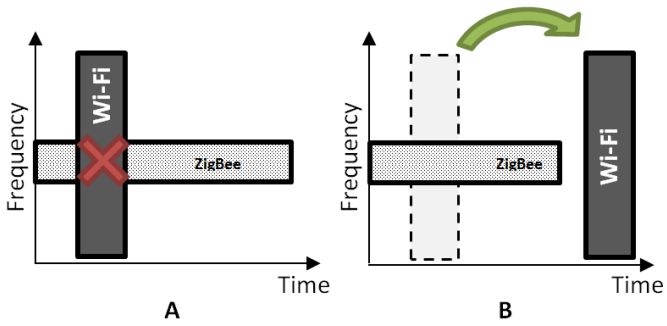


Fig. 2. Interaction between Wi-Fi and ZigBee without (A) and with (B) CACCA

channels and CCA is performed on each channel. While not strictly necessary, we chose to implement this new CCA in a backwards compatible way, complying with the limits imposed on the standard Wi-Fi CCA.

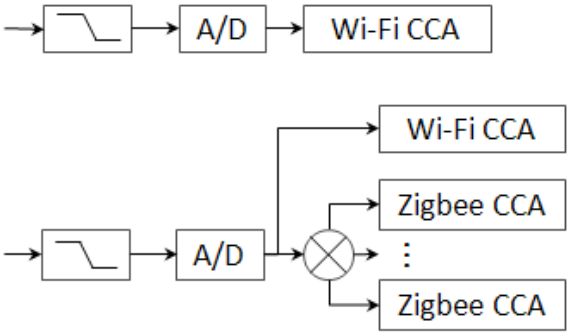


Fig. 3. Standard Wi-Fi CCA (top) and the extended CACCA version (bottom)

2.2 Constraints

Two types of constraints should be considered when a Wi-Fi station is to detect a ZigBee transmission while still conforming to the standard, namely *timing* and detection *sensitivity*.

The Extended Rate PHY (ERP) Wi-Fi standard defines the slot time to be 9 µs [6], consisting of 4 µs of actual channel sensing (CCA Time) and 5 µs for RX – TX turnaround (RxTx_TurnaroundTime). While not strictly necessary, to obtain the highest throughput and limit the additional energy requirements of the CACCA, the implementation should be able to determine the channel state reliably within these 4 µs. Hence not only the CCA itself, but also the signal processing should be finished within the CCA timeframe.

The main purpose of CACCA is avoiding collisions between packets, independent of the technologies used by these packets. Higher level solutions exist to reduce collisions (eg. RTS/CTS, scheduling … [9]) but the cross technology aspect of this problem excludes these solutions. They would require multi technology radios, completely defeating the relative simplicity of CACCA. The CCA sensitivity has as primary target the minimization of the hidden terminal problem with higher sensitivity leading to fewer collisions. CACCA should therefore be capable of detecting signals down to the lowest possible level.

The *timing* and *sensitivity* requirements are in direct conflict with each other as increasing the sensing time improves sensitivity [10]. However, the 4µs limit is a hard limit imposed by the standard and therefore we will strive to achieve the needed sensitivity within the standard defined times.

2.3 Detection Methods

The CACCA requires reliable detection of channel state within the available 4µs CCA time. Multiple methods to perform this detection exist.

Energy detection is achieved by averaging the energy within the channel, without filtering the incoming signal. The channel state is determined by comparing the measured energy level with a predetermined threshold as outlined in figure 4. An energy detector can detect a wide variety of signals with a minimal computational overhead due to the fact that no a priori knowledge about the signal is required. However, the detection sensitivity for a specific technology can be improved through the addition of filtering.

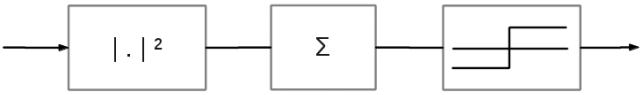


Fig. 4. Simple energy detection architecture

Filtering the received signal with a matched filter before performing the detection (figure 5) will increase the sensitivity [10]. Given complete a priori knowledge of the modulation scheme, this filter corresponds to the receive filter used for demodulation and will provide the best possible detection rate. The modulation scheme used by ZigBee in the 2.4 GHz band is O-QPSK and as such this filter is easily calculated. Not only will the additional filtering increase the processing time, it will also reduce the sensitivity to other signals. While the former reduces the effective sampling time from 4 µs to 3 µs (using a 1 µs filter, figure 6), the latter poses no additional problem for this application.

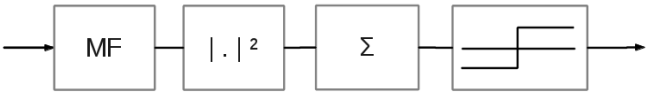


Fig. 5. Simple matched filtering architecture

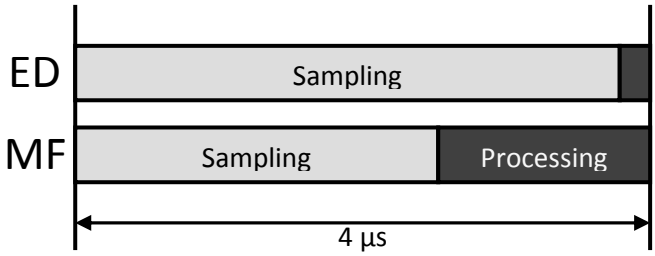


Fig. 6. Sampling and processing periods for energy detection and matched filtering

3 Simulation

The detection methods were simulated in Matlab to verify their performance before they were implemented in hardware. As shown in figure 7, the simulation is split into three parts: signal generation, propagation and the detection algorithms.

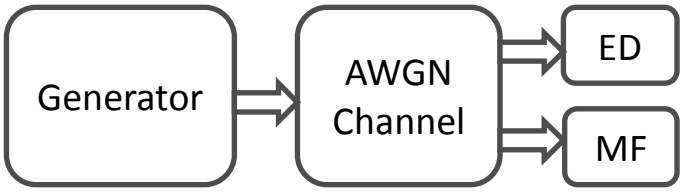


Fig. 7. Simulation overview

The signal generator generates a sample stream containing a ZigBee DSSS (Direct-Sequence Spread Spectrum) signal when the transmitter is active. This stream is passed through an AWGN (Additive White Gaussian Noise) channel model that adds white noise to obtain a predefined SNR (Signal-to-Noise Ratio). Finally, the detection algorithm tries to estimate the original state of the channel state from this noisy signal.

The implementation of the energy detector is fairly straightforward, but the matched filter detector requires the design of an additional filter. This filter is based on the transmit filter described in the IEEE 802.15.4 standard, section 6.5 [7]:

$$p(t) = \begin{cases} sin\left(\pi \frac{t}{2T_c}\right), 0 \leq t \leq 2T_c \\ 0, otherwise \end{cases}, \text{with } T_c = \frac{1}{chip\ rate} = \frac{1}{2\ Mchips/s} = 0.5\ \mu s$$

To obtain the corresponding receive filter, the time reversed complex conjugate of this filter is taken, resulting in the filter shown in figure 8.

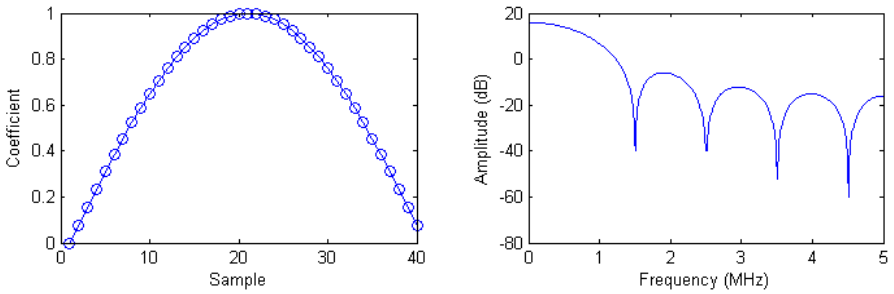


Fig. 8. The matched filter and its frequency response

Detection sensitivity is measured by varying the channel SNR and comparing the output of the detector with the actual channel state. Random 4 μs intervals were sampled by the detector and used to determine channel state.

The detection mechanisms should detect the presence of a ZigBee signal as reliably as possible, albeit keeping the false positive sufficiently low. Without this limit the Wi-Fi transmitter would always sense the channel as busy even though it is free. Therefore we chose the threshold for detection to keep the false positive rate (channel estimated as busy when it is free) below 5%.

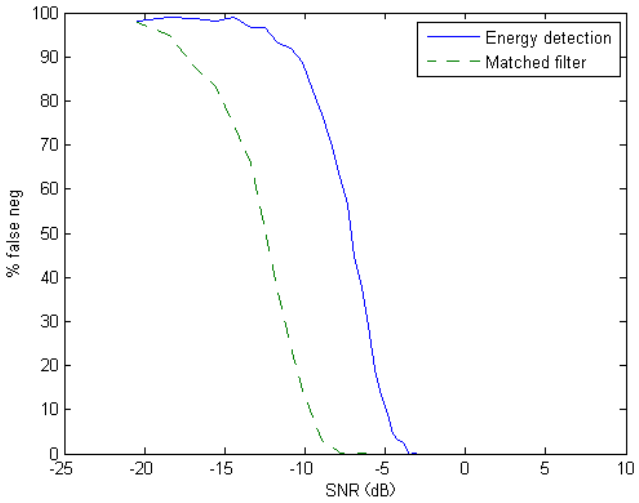


Fig. 9. Simulation results: false negative rate for MF (- -) and ED (—)

The results presented in figure 9 show that both methods are capable of detecting a ZigBee signal successfully. When keeping the false negative rate below 10%, matched filtering is reliable for SNR down to -10 dB, while energy detection is capable of detecting signals with an SNR of -5 dB. Overall, matched filtering provides a gain of approximately 5dB over energy detection, although both methods are suitable for detection.

To verify the correctness of the Matlab implementation, the detection algorithms were also applied to raw data captured from a ZigBee transmission. The processing was done off-line, but we were able to verify that the implemented algorithms are indeed capable of detecting the presence of real signals.

4 Implementation

The very strict timing requirements (section 2.2) mandate a hardware platform that allows both low-level and low-latency access to the RF signal. Several research platforms allow low-level access to the RF signal, but the low-latency requirement rules out host-based platforms like the USRP [11] or SORA [12]. Embedded solutions like the WARP [8] or the Sundance MIMO series [13] do not suffer this drawback and are therefore more suited for this task.

For our implementation the WARP (figure 10) was chosen. This device combines a powerful Virtex-4 FPGA with an RF frontend supporting bandwidths up to 40 MHz in the 2.4 and 5 GHz bands. The reconfigurable logic of the FPGA allows for high speed, low-latency processing of RF signals, while the processor embedded in the FPGA can handle sequential control tasks. Thanks to the tight integration between the processing and control systems, any communication overhead is kept to a minimum and most timing constraints can be met.

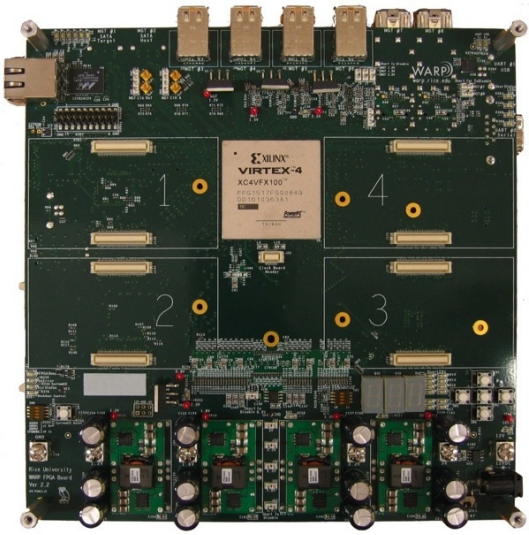


Fig. 10. The Wireless open-Access Research Platform SDR with Virtex-4 FPGA

We originally planned on modifying the CCA of an existing 802.11g implementation to perform CACCA. However, an adequate Wi-Fi MAC for the WARP is lacking so the effort was focused on developing independent detection cores. While a Wi-Fi-like PHY layer is available – the OFDM reference design – extensive modifications

and a new MAC layer would be needed. While this would be an interesting topic in itself, we limited the implementation efforts to the CACCA detection cores. Compared to a complete wireless stack, the complexity of these detection cores will be very limited.

The energy detection and matched filtering algorithms were implemented in the reconfigurable logic of the FPGA as separate cores using the System Generator provided by Xilinx and Simulink. A simple transmit core was also designed to replay prerecorded sample streams.

The final system consists of these cores along with several standard cores from Xilinx and the WARP project. This hardware is connected by the Processor Local Bus (PLB) and a Local Link (LL) connection and is controlled by the software running on the PowerPCs embedded in the FPGA to provide the required functionality:

- Communication with a host PC to control experiments is handled by the serial and Ethernet cores. To support high throughputs the Ethernet core is also connected to the DDR2 memory installed on the WARP.
- RF control is handled by the aforementioned detection and transmission cores. Communication with the actual RF frontend is provided by the WARP radio core that presents an abstracted interface to the other RF cores.
- Additional direct feedback is supplied by several IO cores driving external LEDs.

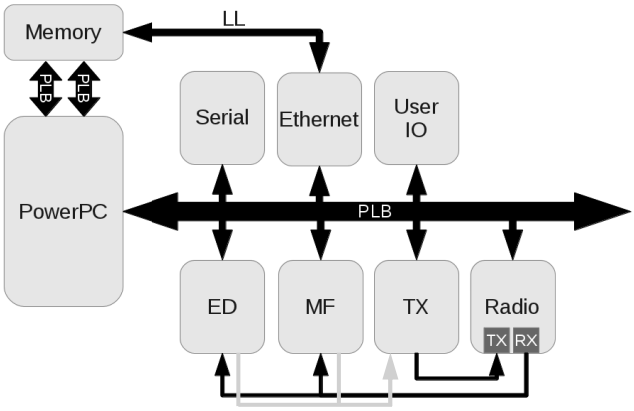


Fig. 11. Architectural overview of the system implemented on the WARP hardware

The complete system (figure 11) is capable of both detecting ZigBee signals using energy detection and matched filtering and interfering with these signals by transmitting pre-recorded samples. While it is capable to function as a stand-alone device, the software running on the PowerPC allows us to reconfigure the system at runtime and display measurement results when performing experiments using an external PC.

5 Experimental Analysis

5.1 Setup

The IBCN research group has access to a controlled RF environment consisting of four shielded enclosures and a network of variable attenuators. Devices to be tested are placed inside these enclosures to minimize external interference and the variable attenuators control the coupling between the enclosures. This setup allows complete control of the RF environment and makes repeatable RF experiments possible. A WARP device was placed in one box and connected to 3 ZigBee nodes in the other boxes through the attenuator network (figure 12). Thanks to the flexible attenuation network this setup can be used to simulate most scenarios involving these devices without moving any hardware.

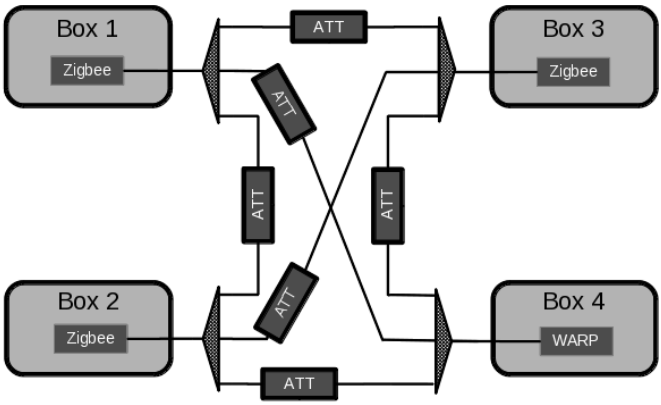


Fig. 12. The wireless test setup at the IBCN research group

5.2 Experiments

Experiments were performed to determine the detection rate and measure the influence of an interferer using CACCA on a standard link.

Sensitivity was measured by varying the attenuation between a transmitting ZigBee node and the WARP. Along with the RF signals, an additional IO signal was routed between the two nodes to indicate the radio state. This line was controlled by the transmitting node and provided the WARP with the state of the radio on the transmitting node. To reduce the influence of timing differences between the radio chip and the microcontroller on the transmitting node, results in a 20 µs interval around a transition on the IO line were discarded. The threshold was again chosen so the false positive rate was below 5%. The configuration of this experiment can be seen in figure 13.

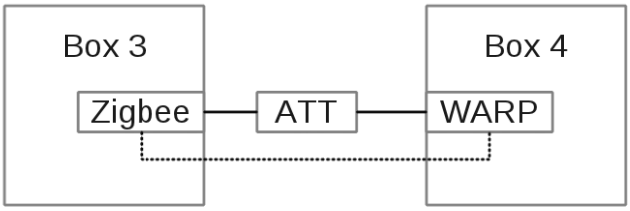


Fig. 13. Sensitivity experiment setup

The effect of CACCA on the goodput of a ZigBee link was measured in a second experiment. Two ZigBee nodes and a WARP were configured in a triangle setup as seen in figure 14: the WARP and the transmitting node were connected directly to the receiving node while a variable attenuator controlled the strength of the signal from the transmitter to the WARP. Interference is generated by the WARP performing CACCA and transmitting short prerecorded Wi-Fi fragments.

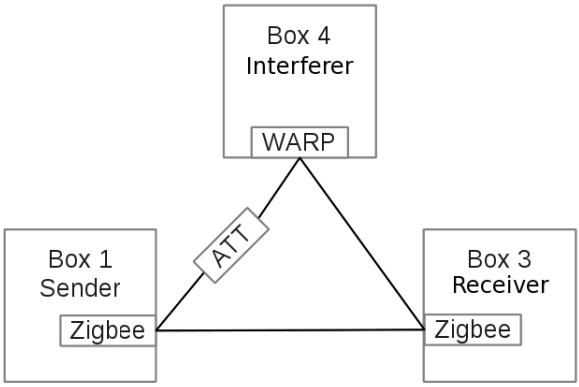


Fig. 14. Goodput experiment setup

5.3 Results

In the first experiment the detection rate of the two CACCA methods was measured by comparing the channel state estimations with the actual channel state. Postulating a 90% reliability, energy detection is capable of reaching this goal for signals down to -79 dBm, while matched filtering can handle signals down to -83 dBm as shown in figure 15. While the differences between both methods are not as big as expected from the simulations, there is still an improvement of about 4 dB.

Comparing these results to the thresholds specified in the Wi-Fi standard we see that both methods perform significantly better. According to the standard the Wi-Fi CCA requires 90% reliable detection of a signal at -76 dBm. For energy detection the improvement is limited to 3 dB, while matched filtering gives a more significant improvement of 7 dB.

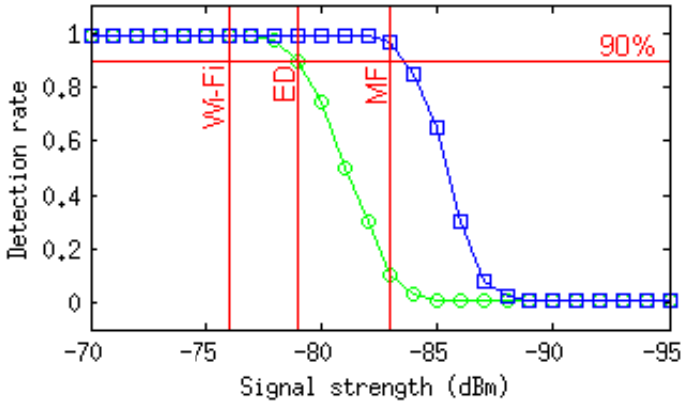


Fig. 15. Detection rate of a busy channel for energy detection (○) and matched filtering (□)

In the ZigBee standard the CCA threshold is specified to be -85 dBm for a sampling time of 128 µs. While reaching this target is not necessary for our case (this implementation is targeted at Wi-Fi systems), comparing the results shows that energy detection is not capable of reaching this threshold in a 4 µs sampling window but matched filtering only falls short by 2 dB.

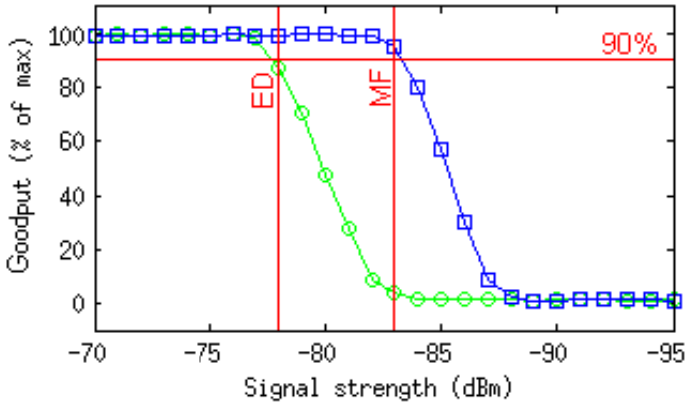


Fig. 16. Goodput for energy detection (○) and matched filtering (□)

In the second experiment the goodput between two standard ZigBee devices is measured while interference is generated by transmitting prerecorded Wi-Fi signals and using different CACCA methods. The results of this experiment are shown in figure 16. When using no CCA, the ZigBee traffic is completely drowned out by the Wi-Fi signal. The goodput is improved by using both CACCA methods: packet error rates < 10% can be achieved using the energy detection method when the signal strength at the interfering Wi-Fi station is above -78 dBm or above -86 dBm when using matched filtering.

6 Future Work

Considering the work presented in this paper, two main areas of improvement can be identified:

The first is *extending* the CACCA to other protocols: the current implementation is focused on the interaction between IEEE 802.15.4 and IEEE 802.11g, but given the plethora of wireless standards and devices, the addition of other technologies like IEEE 802.15.1 will improve coexistence even more.

Secondly, *integration* of the detection cores: due to the low complexity of the detection methods compared to a complete wireless system, it should be possible to include the CACCA in a complete wireless stack. This would allow for additional testing and could possibly lead to an implementation in commodity hardware, as the current hardware is fairly specialized and expensive.

7 Conclusion

Wireless Sensor Networks are increasingly co-located with Wi-Fi, resulting in a decrease of its performance due to an increase in packet loss. Coexistence Aware CCA promises to reduce this packet loss significantly by extending existing CCA methods with methods capable of detecting other technologies. Within this paper we proposed a possible implementation method, implemented a prototype of a Wi-Fi side ZigBee CACCA and experimentally verified its operation.

The main goal of the implementation presented in this paper was remaining backwards compatible with the Wi-Fi standard and keeping the implementation as simple as possible. Therefore we chose the timeframe after which CACCA needs to deliver its channel assessment to be 4 μs, as specified in the Wi-Fi standard. Within this timeframe two possible CCA methods are viable, namely energy detection CCA and matched filtering CCA. With a Matlab simulation we concluded that the ED based approach can reliably detect ZigBee with an SNR down to -5 dB, while MF improves this down to -10 dB.

Both approaches were implemented on the WARP SDR platform, using a combination of software running on the embedded processor and dedicated hardware cores to meet the timing requirements. The original goal of modifying a complete Wi-Fi implementation was abandoned due to the lack of an existing Wi-Fi MAC implementation. Instead Wi-Fi interference was simulated by replaying prerecorded Wi-Fi samples. While this is no optimal solution, it allowed us to perform relevant experiments. The final system is capable of performing ZigBee CACCA and interfering with ZigBee traffic in a Wi-Fi like way.

In a first experiment we verified that the sensitivity of both CACCA methods is indeed better than the default sensitivity specified in the Wi-Fi standard. Compared to the -76 dBm standard, both energy detection (-79 dBm) and matched filtering (-84 dBm) offer significant improvements. A second experiment showed that ZigBee is still able to deliver the maximum throughput when its signal is received stronger than -84 dBm at the Wi-Fi sender. Concluding we can say that this proof of concept clearly shows the benefits of CACCA for technologies operating in the crowded ISM band.

As a final note we would like to address the business opportunities of CACCA. At first sight its business case might seem unclear, for we propose an enhancement to the popular Wi-Fi standard to obtain performance gains for ZigBee devices. However, in [14] we show that due to the relatively low implementation complexity and the more consolidated platform – where one chipset is used in a wide range of devices (ex. PC, laptop, table and phone) – viable business opportunities are feasible.

References

1. Pollin, S., Tan, I., Hodge, B., Chun, C., Bahai, A.: Harmful coexistence between 802.15.4 and 802.11: a measurement-based study. In: Proceedings of 3rd International Conference on Cognitive Radio Oriented Wireless Networks and Communications, pp. 1–6 (2008)
2. Tytgat, L., Yaron, O., Pollin, S., Moerman, I., Demeester, P.: Avoiding collisions between IEEE 802.11 and IEEE 802.15. 4 through coexistence aware clear channel assessment. EURASIP Journal on Wireless Communications and Networking (2012)
3. Thonet, G., Allard-Jacquin, P., Colle, P.: ZigBee – WiFi Coexistence White paper and Test Report, http://www.zigbee.org
4. Yuan, W., Wang, X., Linnartz, J.-P.M.G.: A Coexistence Model of IEEE 802.15.4 and IEEE 802.11b/g. In: 14th IEEE Symposium on Communications and Vehicular Technology in the Benelux, pp. 1–5 (November 2007)
5. IEEE Std. 802.15.1 - 2005, IEEE Standard for Information Technology - Telecommunications and Information exchange between systems - Local and metropolitan area networks - Specific requirements – Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)
6. IEEE Std. 802.11 - 2007, IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications
7. IEEE Std. 802.15.4 - 2006, IEEE Standard for Information Technology - Telecommunications and Information exchange between systems - Local and metropolitan area networks - Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)
8. Rice University WARP Project, http://warp.rice.edu
9. Schiller, J.H.: Medium Access Control. In: Mobile Communications, 2nd edn. Addison Wesley (2003)
10. Tandra, R., Sahai, A.: SNR walls for signal detection. IEEE Journal of Selected Topics in Signal Processing 2, 4–17 (2008)
11. Universal Software Radio Peripheral, http://www.ettus.com
12. Sora Project, https://research.microsoft.com/en-us/projects/sora
13. MIMO 4x4, http://www.rapiddevsys.biz/product_info.php?prod-ucts_id=72
14. Tytgat, L., Barrie, M., Gonçalves, V., Yaron, O.Y., Moerman, I., Demeester, P., Pollin, S., Ballon, P., Delaere, S.: Techno-economical Viability of Cognitive Solutions for a Factory Scenario. In: 2011 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), Aachen, Germany, May 3-6, pp. 182–192 (2001)

Battery State-of-Charge Approximation for Energy Harvesting Embedded Systems

Bernhard Buchli, Daniel Aschwanden, and Jan Beutel

Computer Engineering and Networks Laboratory, ETH Zurich, Zurich, Switzerland
{bbuchli,asdaniel,janbeutel}@ethz.ch

Abstract. Batteries play an integral role in Wireless Sensor Networks as they provide the energy necessary to operate the individual sensor nodes. In order to extend the network's lifetime, and theoretically permit continuous operation even for systems with high-energy consumption, environmental energy harvesting has attracted much interest. It has been shown that the motes' utility can be improved significantly if run-time knowledge of remaining battery capacity is available. In this work, a light-weight and cost effective approach to approximating the battery state-of-charge (*SOC*) based on voltage measurements is presented. Despite commonly perceived as inferior to other approaches, a performance evaluation shows that *SOC* approximations with over 95% accuracy are possible. It is further shown that battery inefficiencies due to *e.g.*, temperature and aging are taken into consideration despite not explicitly modeling these effects. The approach only requires system input voltage measurements, but benefits from optional current and temperature measurements.

Keywords: Wireless sensor networks, battery state-of-charge, modeling, experimentation.

1 Introduction

Batteries play an important role in Wireless Sensor Networks (WSN), which are often deployed at remote locations, *e.g.*, [14], and can therefore not rely on power sources other than batteries. The battery's task is to provide the energy required to operate the WSN motes over extended periods of time. Once the battery is depleted, the sensor node's lifetime has expired, and its battery must be replaced. However, battery replacement can be a time consuming and costly endeavor. Hence, it is imperative that the batteries are reliable and capable of providing the energy necessary to operate the WSN motes according to requirements. Unfortunately, primary batteries can store only limited energy, which is inversely proportional to system lifetime. Particularly for high-energy consumers, the finite energy store imposes reduced utility due to limited runtime. For this reason, rechargeable batteries, supplemented with one of various types of environmental energy harvesting have attracted much interest in WSN scenarios [12,16].

In outdoor WSN deployments, solar energy harvesting with photo-voltaic (PV) energy transducers is most commonly employed. There are a number of reasons for the prevalence of solar energy harvesting. First, solar panels achieve a high energy density, even in indoor scenarios [27]. Low-cost commercial availability and relatively high

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 179–196, 2013.

conversion efficiency [15], reliability, and a periodically recurring energy source – the sun – are further contributing success factors. The cyclic behavior of the sun guarantees periodic opportunities for recharging the batteries. In theory, this enables perpetual system operation, which eliminates the need for periodic and costly battery replacement trips.

It has been shown that awareness of the energy available to the individual motes [21], and the entire network [7] can significantly improve overall system lifetime and utility. However, accurately determining the battery fill level, referred to as battery State-of-Charge (*SOC*), presents a non-trivial task. This is because a battery's *SOC* depends on many battery internal and external factors, such as size and type of battery, the rate at which it is discharged, as well as temperature and battery condition (*e.g.*, age, present *SOC*). While knowledge of the *SOC* may not be mission-critical for some WSN applications, improved observability, predictability, and utility [22, 29] may outweigh the efforts in implementing the necessary functionality.

As discussed in *Section* 2, many existing approaches to determine a battery's *SOC* depend on dedicated hardware to monitor the energy flux into and out of the battery. However, addition of special purpose hardware not only increases design complexity but also system cost [11]. Furthermore, upgrading existing systems with the hardware necessary may not be practical or even possible. Other approaches use complex models that incorporate the non-linear characteristics of batteries under varying and uncontrollable conditions. However, many of the proposed models present significant configuration and computational overhead [28]. Considering the limited computing resources on typical WSN platforms, these models may thus not be feasible for implementation.

This paper presents a practical trace-based, direct-measurement [3] method for on-line battery *SOC* approximation. It is aimed at systems with high energy demands that rely on off-the-shelf photo-voltaic (PV) harvesting components. The proposed approach does not need special purpose hardware, but only requires low-cost sensors that are commonly available on WSN platforms for system health monitoring. The method is computationally inexpensive as it does not depend on complex battery models. Instead, known characteristics of the PV system and the battery's behavior are leveraged.

The contributions of this paper can be summarized as follows. First, a computationally inexpensive, direct-measurement based method for on-line battery *SOC* approximation for energy harvesting systems is presented. Second, the proposed method's performance is evaluated with a proof-of-concept implementation, and experimental results are presented. Despite commonly perceived as inferior [24], it is shown that the direct-measurement approach achieves *SOC* approximation with an average error below 5% when compared to discharge tests. Furthermore, accurate lifetime predictions even with temperature fluctuations and varying battery, and load conditions are possible as the method implicitly accounts for battery inefficiencies like temperature and aging.

The remainder of this paper is structured as follows. *Section* 2 discusses common approaches to battery *SOC* determination, and adaptations to WSN scenarios. *Section* 3 provides an overview of the system architecture and battery characteristics that form the foundation of the proposed approach. *Section* 4 then discusses the *SOC* approximation in detail. *Section* 5 describes the experimental set-up and presents a performance evaluation of the approach. Finally, *Section* 6 presents the concluding remarks.

2 Review of Battery State-of-Charge Approximation

Efforts to estimate the residual charge contained in a battery have been made for almost as long as rechargeable batteries exist, and increasing popularity of hybrid electric vehicles has further pushed this research topic. Hence, literature review reveals a host of battery state-of-charge (*SOC*) approximation models, of which common techniques are presented in this section. A brief review of WSN specific approaches is also provided.

State-of-Charge Determination Methods. One of the earliest approaches is in the form of current integration, usually referred to as coulomb counting [24,25]. With this approach, charge flowing from (in case of discharging), or to the battery (charging) is measured and integrated over time. After subtracting the net charge from a full battery, the residual charge contained in the battery can be obtained. Although widely used today, this approach suffers from a number of issues. First, to yield acceptable accuracy, battery inefficiencies that are not directly measurable must be compensated for. Second, inaccurate current readings lead to an accumulation of error, but accurate current measurements are expensive, and the addition of the necessary hardware can add substantial design complexity and development cost. Finally, since the performance depends on known initial conditions, regular re-calibrations are required [24,25].

A technique that attempts to improve the performance of coulomb counting is called book-keeping [3]. With this approach, common battery inefficiencies, such as discharging efficiency, self-discharge, and capacity loss are taken into consideration to yield a more accurate *SOC* indication. These inefficiencies, and details on how the model proposed in this paper copes with their effects are discussed in *Section* 5.

A discharge test under controlled conditions is considered to be the most reliable means of approximating a battery's *SOC* [24], and commonly performed by the battery manufacturer [9]. For this reason, results from discharge tests are used as ground-truth for the evaluation of the proposed *SOC* approximation method. Nevertheless, discharge test are not feasible for most if not all practical applications because system operation is interrupted willingly and possibly for long periods of time.

Another technique that is applicable to vented lead-acid (LA) batteries with liquid electrolytes is in the form of measuring the electrolytes' physical properties. LA batteries, which are common in cars and stationary photo-voltaic energy harvesting systems, exhibit a relationship between the electrolyte's properties and *SOC*. Therefore, by measuring *e.g.*, specific gravity, conductivity, and ion-concentration, the battery's *SOC* can be determined. However, to obtain an accurate *SOC* indication, the temperature must be considered, and measurements may only be taken after a proper charging cycle and appropriate resting period of the battery. This, together with the need for measuring the battery's internal properties limit the applicability of this approach.

A lead-acid battery's electrolytes' properties are also directly related to the open-circuit voltage. Hence, the battery's *SOC* can be obtained by measuring the battery's terminal voltage when no load is connected. However, this necessitates periodic load disconnects, which requires special design considerations. Just like with the previous approach, appropriate resting times are required to obtain meaningful measurements.

Recently, there has been significant efforts in devising analytic models for batteries and *SOC* determination. An excellent review is given in [28]. These models range from electrical circuit equivalents, over stochastic load modeling, to mixed models that

incorporate both experimental data and physical laws. In particular, extended [30], and adaptive [13] Kalman filtering techniques have emerged. Unfortunately, these approaches require significant configuration efforts and exhibit computational complexity that preclude application in low-performance systems such as WSN motes [28].

Finally, the approach most closely related to the one discussed in this work consists of direct voltage measurements [25]. Voltage measurements are generally considered to yield inaccurate *SOC* indication because a battery's voltage profile depends on discharge rate, temperature effects, and age of the battery to name a few. However, as will be elaborated in the remainder of this work, if the battery's behavior under load, and the discharge current are known (or measurable), the *SOC* can be obtained with good accuracy even under varying battery operating conditions.

State-of-Charge Determination in WSN Scenarios. Of the approaches discussed so far, only a subset is feasible for implementation in WSN scenarios. This may be due to cost considerations, physical constraints, and limited processing power available on the motes. In addition to special-purpose hardware [17,26], fully software [8,11,18] based approaches to *SOC* determination have been proposed, and are briefly introduced here.

In [8] a software based *SOC* model for lifetime prediction is presented. The platform load is characterized by a constant average current for each activity (*i.e.*, sensing, processing, and transmission). The *SOC* is then approximated in software by "counting" the charge over the period of time a given component is active. The authors report a lifetime estimation error below 10%. However, their approach requires current measurements of the PV system or, alternatively a light meter. Both require pre-deployment design considerations, which precludes its use in existing systems.

Although not aimed at energy harvesting systems, [11] and [18] follow a similar approach. Software routines are implemented that are executed every time a certain hardware component is switched on or off. The total time a given component is active is then multiplied by its average current drain, which is obtained by pre-deployment power profiling of the system. Finally, to obtain the overall system energy consumption (and hence infer the battery *SOC*), all components are summed up. It is not clear what accuracy either of these approaches achieves, but considering that coulomb counting integrated circuits (ICs) require frequent re-calibration, it can be assumed that the error in purely software counting techniques accumulates just as rapidly.

In contrast to purely software based approaches, [26] presents Heliomote, a custom energy harvesting hardware module as add-on for Berkley/Crossbow motes. It is a complete power management solution that autonomously controls solar harvesting, charging of the battery, storage and power routing, and provides harvesting and battery state information to the host platform. While not exclusively aimed at providing *SOC*, the authors show that harvesting aware power management can improve system utility.

Similarly, SPOT [17] is another application specific custom micro power meter for energy monitoring of MicaZ motes. As motivation, the authors state that commercially available ICs are not designed to meet WSN application requirements. Similar to Heliomote, the immediate aim is not battery *SOC* determination, but rather empirical on-site evaluation of low power designs at scale. A price tag of $25 is stated, but it is not clear how much re-design and integration effort would be necessary to adapt the SPOT module to other mote platforms.

3 System Concept

System Architecture and Assumptions. For the proof-of-concept implementation discussed in this paper, an off-the-shelf (OTS) energy harvesting set-up depicted in *Figure 1* is assumed. It consists of an energy harvesting module, such as a solar panel, a PWM charge controller [20], and one or multiple Valve Regulated-Absorbent Glass Mat (VR-AGM) sealed lead-acid batteries [9]. While this is not a typical set-up for low-power motes with simple sense-and-transmit applications it can be considered a reasonable, low-cost set-up for systems with high-energy consumption due to high-power sensors and increased duty-cycles. Aside from WSN basestations [14] that usually have considerable energy demands, recently proposed application scenarios like continuous GPS [6], acoustic emission [31], or Audio/Video surveillance [2] validate such a set-up.

With an OTS set-up, the charge controller regulates proper charging of the battery, but does not provide *SOC* information. Therefore, the proposed approach aims at approximating the *SOC* in a battery type, and set-up independent manner. The goal is to provide a solution that does not depend on extensive hardware support, but gets by with low-cost sensors commonly available on contemporary WSN motes. Furthermore, accurate *SOC* indication should be possible without requiring extensive processing.

Although a lead-acid (LA) battery chemistry is assumed in this work, the approach is expected to apply to any battery chemistry with an appropriate discharge profile. The model is agnostic to a particular set-up and harvesting source, and only requires a well-defined behavior of the charge-controller and the battery behavior under load.

Battery Model. Devising an analytic battery model that takes into consideration all the battery's non-linearities has proven to be complex [3] and, depending on application, computationally prohibitively expensive [28]. However, incorporating all inefficiencies into the model may not be necessary for achieving acceptable *SOC* approximation. Instead, from the illustration of a battery's voltage profile (obtained from its data-sheet [9]) in *Figure 2a* it is observed that a battery exhibits qualitatively very similar voltage versus Depth-of-Discharge (*DoD*) curves for different discharge rates (*DoD* is the complement of *SOC*). The battery's charging and discharging characteristics are explained in the following. *Section 4* then explains how these characteristics are leveraged to obtain information on the battery's state-of-charge.

Capacity and Cut-off Voltage. The manufacturer provided capacity rating specifies the battery's nominal capacity, which refers to the maximum charge that can be withdrawn before fully depleting the battery (*i.e.*, 100% *DoD*) at a specified discharge rate and temperature. Usually, the discharge rate and temperature are assumed to be C/20

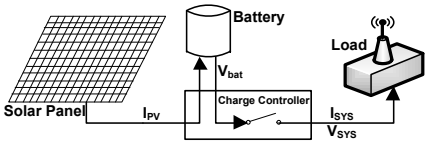
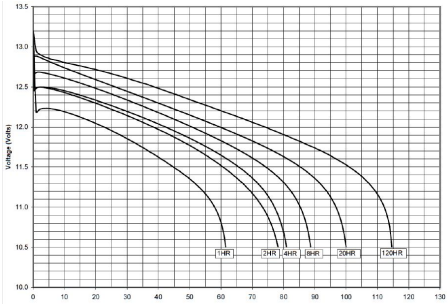
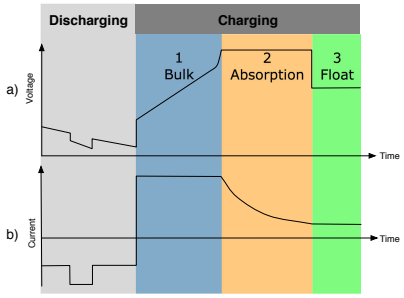


Fig. 1. System architecture with off-the-shelf energy harvesting components (solar panel, battery, charge controller), and load. I_{sys} and V_{sys} are assumed to be observable.



(a) Manufacturer provided Depth-of-Discharge (*DoD*) versus terminal voltage [9] for 1, 2, 4, 8, 20, and 120-hour discharge rate (as a percentage of 20-hour rate) of a VR-AGM battery.



(b) Relation between battery terminal voltage (top) and current (bottom) during the charging and discharging processes.

Fig. 2. (a) Battery discharge profile at different discharge rates, and (b) qualitative illustration of battery behavior during charging and discharging processes

and 25 °C, which means that the battery is fully discharged in 20 hours. However, since deep discharge cycles cause irreversible chemical reactions, the battery should not be fully discharged. To protect the battery and maximize its lifetime, commercial charge controllers therefore implement a cut-off voltage, below which the load is disconnected until the battery recovers to a certain *SOC* (usually 60% of nominal capacity).

The known cut-off voltage provides two pieces of information that are leveraged in the proposed model. First, the cut-off voltage is used to define the 0% *SOC* (or 100% *DoD*) of the battery. Second, since this voltage is set such that it limits the *DoD* to 20%, it can be assumed that the battery can deliver 80% of the nominal capacity. This assumption has been experimentally verified with a coulomb counter [19] in *Section* 5.

Charge and Discharge Profiles. The behavior of a battery can be classified into two processes: charging and discharging. The charging process can be further segmented into three phases: Bulk, Absorption, and Float charging. These processes are illustrated in *Figure* 2*b* and briefly explained in the following. For simplicity, ideal conditions – that is no charging inefficiencies, and a stable power source – are assumed.

Discharging. The battery is discharged with a certain drain current that defines the rate of discharge. Electro-chemical reactions lead to an increase in battery internal resistance, which causes a drop of the battery's terminal voltage directly related to the magnitude of the drain current, as illustrated in *Figure* 2*a*.

Bulk Charging. In this phase, a constant current is applied, which causes the battery potential to increase until a preset, temperature dependent voltage is reached. The time for reaching this voltage depends on the magnitude of the drain current and the battery's *SOC* at the beginning of the charging process.

Absorption Charging. In this phase the voltage is kept at a constant voltage level defined by the bulk phase. The charging current is tapered off until it drops below a certain percentage of the battery's nominal capacity (0.5% for the batteries used [9]), after which the battery can be considered fully charged.

Float Charging. The final charging phase attempts to keep the battery at 100% *SOC*. To prevent overcharge, the voltage is reduced to a temperature dependent float voltage and the charging current is kept constant. Depending on the present *SOC*, and the magnitude of the charging and drain currents, this phase may not be reached.

4 State-of-Charge Approximation

This section develops a set of equations that can be used to obtain harvesting source, and set-up independent indication of residual charge stored in the battery.

4.1 Discharge Approximation

To model the behavior of the battery under varying loads, a three step procedure is performed to extract the necessary information. The following explains the procedure for the discharging process. *Section 4.2* then elaborates on the charging process.

Trace Generation. A trace consists of fixed interval measurements of the battery voltage and system drain current until the battery is fully drained. The magnitude of drain currents should be chosen such that the traces bound the expected operational range of the system. An illustration of discharge traces at various rates is shown in *Figure 2a*. Since the data provided by the manufacturer is for rates much higher than required, the battery is profiled with discharge rates appropriate for the particular system [5] used.

Trace Transformation. Next, a domain transformation of the traces from the time domain to Depth-of-Discharge (*DoD*) domain is performed, which is defined for the interval [0,1]. To this end, the known cut-off voltage of the charge controller is leveraged to define 100% *DoD*. The known battery voltage of a fully charged battery defines 0% *DoD*. Using the recorded time t_{cutoff} it takes the battery to reach the cut-off voltage, the *DoD* is assigned linearly in time ($DoD(t) = t/t_{cutoff}$). Finally, the traces are inverted to obtain *DoD* as a function of battery voltage.

Coefficient Extraction. After transformation of the traces, the battery voltage dependent *DoD* can now be approximated with a polynomial function of order n (*Equation (1)*). The choice of n is a trade-off between computational complexity and fitness of the traces, but it has been found that a quadratic approximation achieves good results (see *Section 5*). Since the coefficients a_n vary with the load, they depend on the discharge current. Rather than using the absolute drain current, it is noted that the load caused by the drain current is relative to the nominal capacity of the battery. For this reason, the relative load (RL) is introduced, which normalizes the discharge current I_{sys} with respect to the battery's nominal capacity C_{bat}, and is defined as $RL = I_{sys}/C_{bat}$. Thus, the *DoD* approximation for trace i is defined by *Equation (1)*.

$$\widehat{DoD_i}(V_{bat}, RL) = a_{i,n}(RL)\cdot V_{bat}^n + a_{i,n-1}(RL)\cdot V_{bat}^{n-1} + \ldots a_{i,1}(RL)\cdot V_{bat} + a_{i,0}(RL) \quad (1)$$

It is desirable to obtain a single *DoD* equation that covers the entire operating range, including relative loads other than the ones recorded. Therefore, to obtain the coefficients $a(RL)$ in *Equation (1)*, another set of polynomial approximations of order m is

performed over all trace coefficients $a_{i,n}$ to find the coefficients $b_{n,m}$ in *Equation* (2). The end result is a single equation representing the *DoD* as a function of relative load and battery voltage, and $n+1$ equations for the the coefficients a_n (*Equation* (2)).

$$a_n(RL) = b_{n,m} \cdot RL^m + b_{n,m-1} \cdot RL^{m-1} + \ldots + b_{n,1} \cdot RL + b_{n,0}, \quad m \geq n \quad (2)$$

4.2 Charge Approximation

To obtain a model for the *SOC* approximation during the charging process, a procedure similar to the one discussed in *Section 4.1* is followed. However, as an off-the-shelf set-up is assumed, the charging current, I_{PV}, produced by the solar panel cannot be measured directly and must therefore be approximated. This is necessary because, depending on the magnitude of I_{PV}, and assuming non-zero drain current I_{sys}, the resulting charge (I_{bulk}, I_{abs}) flowing into the battery may be positive or negative.

Bulk Approximation. In the ideal case, the bulk charging current, $I_{bulk} \leq I_{PV}$, flowing into the battery is constant, causing the voltage to increase monotonically. The relative charging current, $RC_{bulk} = I_{bulk}/C_{bat}$ is used to determine the current *SOC* during the bulk phase in a conservative way (conservative because the current is relative to nominal, rather than actual capacity). If charging inefficiencies are ignored for the moment, this increase is relative to the previous *SOC*. Hence the *SOC* is computed using *Equation* (3), where t_{bulk} represents the time elapsed since the last update.

$$\widehat{SoC}(t) = \widehat{SoC}(t - t_{bulk}) + \widehat{RC}_{bulk} \cdot t_{bulk} \quad (3)$$

However, the exact value for I_{bulk} is unknown, and $\widehat{RC}_{bulk}$ must therefore be approximated. Fortunately, the rate at which the voltage increases during the bulk phase is directly proportional to $\widehat{RC}_{bulk}$. Therefore, the slope $\frac{\delta V}{\delta t}$ of the trace is approximated by a linear regression. Since the traces are collected with known relative charging current, the corresponding slopes of the voltage profiles are used to extract the coefficients c_0 and c_1 of the first order approximation of $\widehat{RC}_{bulk}$, shown in *Equation* (4).

$$\widehat{RC}_{bulk} = c_1 \cdot \frac{\delta V}{\delta t} + c_0 \quad (4)$$

Absorption Approximation. As explained in *Section 3*, the absorption phase is characterized by an exponentially decreasing charging current with initial value given by $\widehat{I}_{bulk}$. Due to constant terminal voltage during this phase, I_{abs} cannot be approximated by the voltage in the same way it is done for bulk charging. Since it is assumed that the charging current I_{PV} cannot be measured, one option to approximate the relative charge $\widehat{RC}_{abs}$ is by considering the known behavior of I_{abs}. For this an ideal scenario is assumed, *i.e.*, the current source does not fluctuate significantly.

Therefore, $\widehat{RC}_{abs}$ is approximated with *Equation* (5), where $\widehat{\lambda}$ is the decay constant, which is dependent on the relative charging current during the bulk phase. The true decay constant λ is found by an exponential fit of the corresponding charging traces. Then, as *Equation* (6) shows, $\widehat{\lambda}$ is approximated by a linear interpolation of the relative charging current $\widehat{RC}_{bulk}$. The coefficients d_1 and d_0 are obtained by a linear regression of λ and RC_{bulk} for each trace. Finally, since the relative charging current RC_{abs} is

decreasing over time, the *SOC* is updated differentially for each discrete time step Δt (*Equation* (7)). *Equation* (5) can be approximated by a Taylor series if need be.

$$\widehat{RC}_{abs}(t) = \widehat{RC}_{bulk} \cdot e^{(\hat{\lambda}(\widehat{RC}_{bulk}) \cdot t_{abs})} \tag{5}$$

$$\hat{\lambda}(\widehat{RC}_{bulk}) = d_1 \cdot \widehat{RC}_{bulk} + d_0 \tag{6}$$

$$\widehat{SoC}(t) = \widehat{SoC}(t - \Delta t) + \widehat{RC}_{abs}(t) \cdot \Delta t \tag{7}$$

Float Approximation. The *SOC* of the float phase is by definition 100%. This phase can easily be detected by the float voltage given in the battery's data-sheet [9].

4.3 Battery State Tracking

For tracking the behavior of the battery over time, a Finite-State Machine (FSM) is defined. The FSM's states correspond to the 4 charge and discharge phases explained in *Sections* 3 and 4.1. An additional UNKNOWN state is introduced to capture initial conditions. *Figure* 3 shows the states and transitions of the FSM used. The triggers for the individual state transitions are listed in *Table* 1.

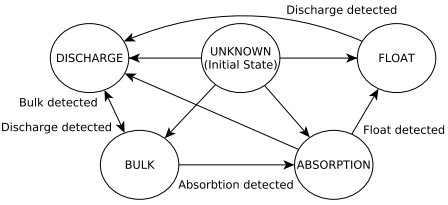


Fig. 3. State Machine for charge/discharge tracking. Transitions are listed in *Table* 1.

Table 1. State transition matrix. $\uparrow$ and $\downarrow$ refer to 'increase' and 'decrease', respectively, while $\not\uparrow$ and $\not\downarrow$ refer to 'no increase' and 'no decrease' of the respective parameter.

		Next State				
		UNKNOWN	DISCHARGE	BULK	ABSORPTION	FLOAT
Current State	UNKNOWN	No change	$V \downarrow, I_{sys} \not\uparrow$	$V \uparrow, I_{sys} \not\downarrow$	$V \geq V_{abs}$	$V \geq V_{float}$
	DISCHARGE	–	No change	$V \uparrow, I_{sys} \not\downarrow$	$V \geq V_{abs}$	$V \geq V_{float}$
	BULK	–	$V \downarrow$	No change	$V \geq V_{abs}$	$V \geq V_{float}$
	ABSORPTION	–	$V < V_{float}$	–	No change	$V_{float} \leq V < V_{abs}$
	FLOAT	–	$V < V_{float}$	–	–	No change

4.4 Measurement Considerations

The goal is to implement a light-weight HW/SW approach to battery *SOC* approximation that does not depend on special-purpose hardware. Nevertheless, since the approach relies on voltage measurements, at least a voltage sensor is required that can measure the unregulated system input voltage with sufficient resolution and accuracy. The drain

current can be obtained by profiling, as is done in [8, 11, 18]. However, the instantaneous *SOC* approximation can be improved by employing a current sensor to measure the true system drain current. If the *SOC* should also be tracked during charging of the battery, a temperature sensor to measure the battery's temperature must also be available because the charging process is temperature sensitive. These sensors are generally present on WSN motes for health monitoring, and are therefore not considered special-purpose hardware.

During discharging of the battery, the proposed algorithm is stateless and only considers the present measurements to compute the *SOC*. This means that the model's accuracy is independent of the sampling period, and the health measurements of the respective application can be conveniently re-used whenever they are scheduled anyway. This eliminates measurement overhead and reduces possible interference with normal system operation. However noise on the sensor readings may affect the solution quality. For this reason an exponentially weighted moving average (EWMA) filter with length 10 is used to smoothen all physical measurements.

For the charging process the output depends both on the present measurement and previous *SOC*. In this case it is clearly beneficial to use a sampling period appropriate to capture the dynamics of the energy source used. With solar harvesting the source tends to fluctuate slowly, so a sampling period of a few minutes can be considered sufficient.

With an off-the-shelf set-up assumed and depicted in *Figure 1*, the system input voltage does not correspond to the actual battery voltage. However, to approximate the *SOC* in a set-up independent manner, the battery's terminal voltage is required. Therefore, voltage drops caused by the charge controller and the power cable, or any other consumers must be explicitly accounted for. Finally, it is worth noting that for this algorithm no *SOC* updates can be made during sleep periods, unless supervisory circuit with the ability to read the sensors and execute the FSM introduced in *Section 4.3* is available.

5 Evaluation

This section presents the performance result of the proposed *SOC* approximation approach when used with an OTS power sub-system and high energy consumer, *e.g.*, WSN basestation [14] or applications with high-power sensors and increased duty-cycles [6].

5.1 Experimental Set-Up

Model Parameters. In *Section 4* general equations for obtaining the *DoD* as a function of the battery voltage and relative load have been developed. *Table 3* list the respective coefficients, which have been found by fitting discharge traces of a single battery identical to *BAT1* (*Table 2a*) with relative loads of 0.00471, 0.01194, and 0.01638 with a quadratic approximation. A cubic fit was found not to improve the model's performance appreciably.

Test Set-up. For evaluating the proposed model, constant power discharge tests are carried out with three Absorbent Glass-Matt (AGM) batteries [9] listed in *Table 2a*.

A custom platform of which details are given in [5], presents the load and executes the logic described in *Section 4* with the parameters shown in *Table 2b*. To verify that batteries of the same size yield the same results, a discharge test is performed on two identical batteries. The differences are found to be insignificant, hence only one set of results is presented in the following.

Evaluation Baseline. To validate the assumption about available battery capacity made in *Section 3*, an external coulomb counter [19] is placed between the charge controller and the battery. In order to evaluate the method's ability to adapt to conditions that are not explicitly modeled, such as varying size, condition, etc., a number of discharge tests are carried out and discussed in the following. Discharge tests under controlled conditions are considered to yield accurate indication of *SOC* [9, 24]. Before starting each test, the respective battery is fully charged and allowed to rest for 4 hours.

Table 2. (a) Type, capacity, and condition of the batteries [9] used, and (b) Sampling rate and filter specification for smoothing of physical measurements

Name	Type	Capacity (C/20)	Condition
BAT0	GPL-1400T	43 Ah	Old
BAT1	GPL-U1T	33 Ah	New
BAT2	GPL-1400T	43 Ah	New

Description	Value
Sampling rate	30 sec.
Filter type	EWMA
Filter length	10

Table 3. Coefficients for *Equations* (2), (4), and (6) found by interpolation of traces

Coeff.	Value	Coeff.	Value	Coeff.	Value
$b_{2,0}$	-8.321e-9	$b_{1,0}$	-0.07018	$b_{0,0}$	100
$b_{2,1}$	-0.002257	$b_{1,1}$	4.492	$b_{0,1}$	0
$b_{2,2}$	0.08133	$b_{1,2}$	-204.7	$b_{0,2}$	0

Coeff.	Value
c_0	0.005573
c_1	1.296

Coeff.	Value
d_0	-0.003891
d_1	-0.1091

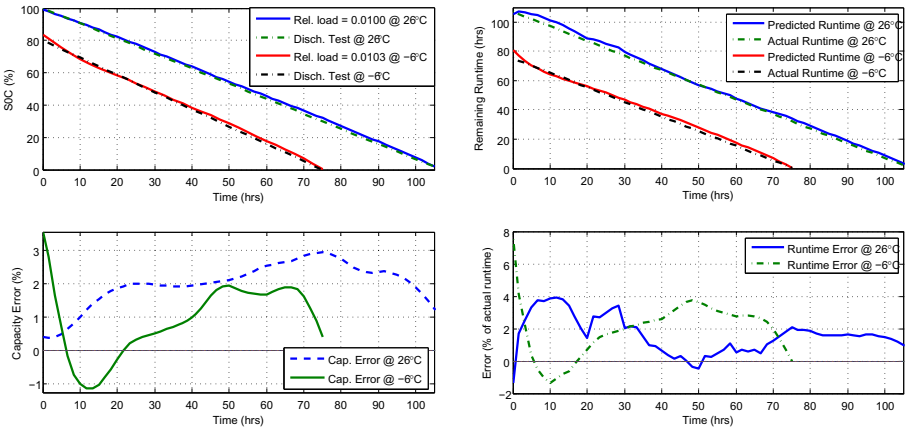
5.2 Experimental Results

In the discussion so far, battery inherent inefficiencies have been ignored. However, when operated under varying, and uncontrollable conditions, batteries exhibit non-linearities that cannot be neglected. Therefore, the performance of the *SOC* approximation is first evaluated with a battery exposed to different temperatures. Then, the model's ability to adapt to aging batteries is investigated. Finally, test results obtained with varying loads, and the model's accuracy in predicting remaining runtime are presented. It is important to note that the goal is to evaluate the performance of the proposed approach, and *not* the battery behavior under the conditions mentioned.

Temperature Effects. Temperature variations have a strong impact on the battery's *SOC* as they affect the apparent capacity [9, 28]. Low temperatures cause a reduction of the electrochemical activity, which leads to a (temporary) reduction of energy available (relative to the rating), and the opposite is true for high temperatures. Since the battery parameters were obtained at 22 °C, the first experiment evaluates the model's ability to

adapt to lower temperatures. To this end, *BAT1* is cooled down and kept at $-6\,^{\circ}\mathrm{C}$ while being discharged with a constant average relative load of 0.0103. This specific temperature is selected as it represents a typical average temperature during winter months at a particular deployment location of interest [4]. *Figure 4* and *Table 4* show the results, which are briefly discussed in the following.

The upper graph in *Figure 4a* shows the *SOC* approximation of both the model and the ground-truth at constant $-6\,^{\circ}\mathrm{C}$, and for reference at $26\,^{\circ}\mathrm{C}$. The lower graph shows the capacity error, which is defined as $SoC_{model} - SoC_{DT}$, *i.e.*, the difference between the *SOC* indicated by the model and capacity inferred by the discharge test. At $-6\,^{\circ}\mathrm{C}$ the maximum overestimate of 3.54% occurs only in the very beginning but quickly drops during the first 6 hours until reaching a maximum underestimate of -1.17% after 12 hours. After 25 hours, and for the remainder of the test, the error varies between roughly 0.3% and 2% with a mean deviation from the ground-truth of 0.85%.



(a) *SOC* approximation (top) and error (bottom).

(b) Runtime prediction (top) and prediction error (bottom).

Fig. 4. Constant power discharge test at constant $26\,^{\circ}\mathrm{C}$ and $-6\,^{\circ}\mathrm{C}$ with battery *BAT1*

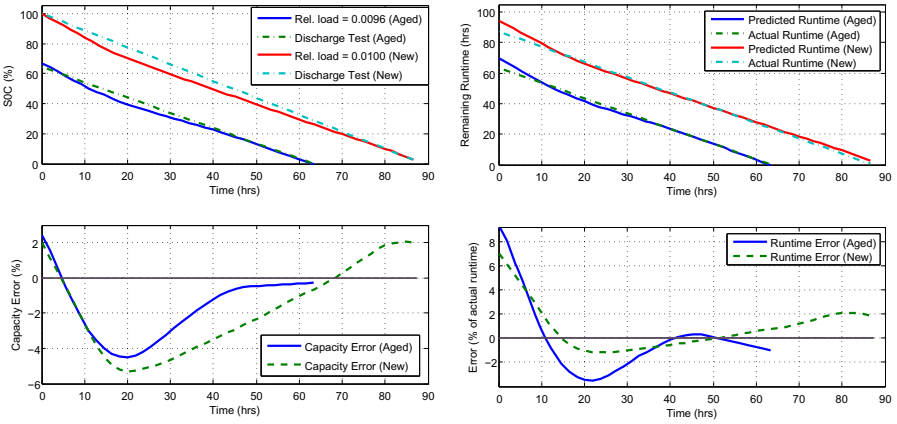
The reason for the initial overestimate is because of the increased rate at which the voltage drop occurs due to the low temperature. This causes the model to overestimate the *SOC* until the battery has adjusted to the load, at which point the rate of change in the voltage profile slows down and reflects the actual condition. The consistent overestimate in both cases is likely due to considering too little voltage drop across the charge controller and cabling (see *Section 4.4*).

Another inefficiency that is highly temperature dependent and common to all batteries is referred to as self-discharge. This phenomenon causes the battery to discharge at a type and temperature specific rate [1] even with no load connected. This effect is ignored as it is assumed that an off-the-shelf charge controller presents a load that is always larger than the battery's self-discharge.

In summary, the temperature, and – as will be shown in the following – aging effects are indirectly considered through changes in terminal voltage that occur due to

temperature variations without a corresponding change in drain current. The change in voltage induced by these effects causes the model to yield a correspondingly lower or higher *SOC*. It is important to note that 0% *SOC* is not affected by temperature because the charge controller disconnects the load at a temperature independent cut-off voltage.

Aging Effects. Each charge/discharge cycle causes irreversible chemical reactions within the battery. This causes a gradual decrease in the maximum charge the battery can store and deliver [3], and is referred to as capacity loss. Conventionally, a battery has reached its end-of-life when it fails to store and deliver a certain percentage (usually 80%) of its nominal capacity [9]. Since capacity and voltage are related [24], an old battery will exhibit a lower terminal voltage (compared to a new battery) due to lower electrochemical activity. Therefore, identically to temperature effects discussed previously, the model indirectly considers aging effects via its impact on terminal voltage.



(a) *SOC* approximation (top) and error (bottom).

(b) Runtime prediction (top) and prediction error (bottom).

Fig. 5. Discharge test with aged (*BAT0*), and new (*BAT2*) batteries

Figure 5 and *Table* 4 show the results of the discharge test with *BAT0*, which has reached its end-of-life. For reference, the same discharge test is carried out with *BAT2*, which is of the same type, but in new condition. The initial capacity is determined using a coulomb counter, which indicates that only 52% of the nominal capacity (equivalent to roughly 65% of available capacity) can be withdrawn when *BAT0* is fully charged. As is evident from the top most graph in *Figure* 5a, this agrees very well with the initial *SOC* approximated by the model. The maximum and mean deviation from the ground-truth of -4.49%, and -1.69% respectively may seem rather high. However, the negative values represent an underestimate, which is maintained almost over the entire range. This may be viewed as a conservative indication of *SOC*, and, depending on application, may even be desirable. It is worth noting that, except for in the beginning, both discharge tests yield very similar profiles, which validates the method's applicability to varying battery conditions. Unfortunately, the model parameters, which are obtained with a battery size

equivalent to *BAT1*, tend to introduce error when used with the larger batteries. The effect is visible for both cases by the trough shortly after starting the test.

Load Variations. Practical applications typically do not exhibit constant power dissipation, hence this section presents the model's performance when exposed to varying drain currents. Load changes frequently occur in real systems, due to *e.g.*, duty-cycling. It is well known [10,23] that a battery behaves non-linearly when discharged at different rates. These rate dependent effects are known as rate-discharge, and recovery effects.

Figures 6 and 7 and *Table 4* present the results of a discharge test with bimodal load changes. As illustrated in the lower graph in *Figure 6*, which shows the voltage profile, the high-power mode presents a maximum relative load of 0.0170 and is active 35% of the time, while the "low-power" mode presents a relative load of 0.0055 for the remaining 65%. In this discharge test, the rate discharge and recovery effects become visible. With variations in the load, the battery is allowed to recover some of the charge regularly. Unfortunately, the model tends to continuously fluctuate between under- and overestimating the *SOC*. Towards the end of the discharge test, the maximum overestimate reaches 4.98%. Nevertheless, despite the load changes and their visible effects on the voltage profile, the *SOC* exhibits a relatively linear tendency with a mean deviation from the expected value of 3.02%.

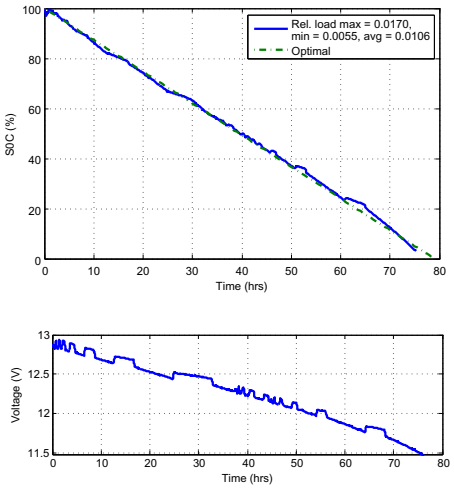


Fig. 6. *SOC* approximation (top), and voltage profile (bottom) for bimodal load

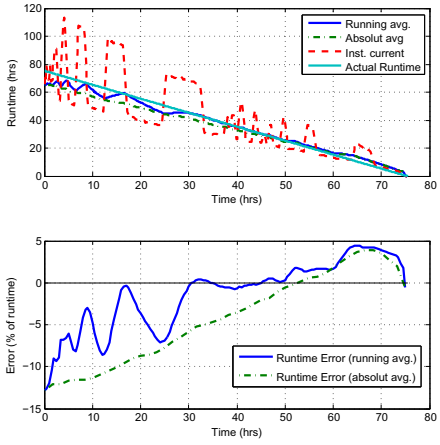


Fig. 7. Runtime prediction (top) and prediction error (bottom) for bimodal load

Lifetime Estimation. The ability to predict the remaining runtime is clearly more beneficial to the system than mere indication of battery *SOC*. For this reason, this section investigates if the approximated *SOC* can be used to easily predict remaining runtime. In addition to the instantaneous *SOC* and drain current, the manufacturer specified nominal capacity C_{bat} in mAh is used as basis for computing the remaining lifetime L, as shown in *Equation (8)*.

$$L(t) = SOC(t) \cdot \frac{C_{bat}}{I_{sys}(t)} \cdot \alpha \tag{8}$$

The scaling factor α is required to account for rate-discharge and recovery effects with variable loads. Therefore, All but the experiment with varying loads use $\alpha = 1$. This implies that when the battery is discharged with the constant loads evaluated, the assumed capacity (*i.e.*, 80% of C_{bat}) is actually available. In the more interesting case with varying loads, rate-discharge and recovery effects have an impact on the assumed capacity, which requires an appropriate scaling to yield acceptable lifetime prediction. For the results shown in *Figure 7*, $\alpha = 0.7$ has been found to yield the lowest average lifetime prediction error.

As expected, however, α does not behave linearly with the discharge duty-cycle. Another set of tests carried out with a duty-cycle of 50% and periods of one ($\alpha = 0.85$), two ($\alpha = 0.88$), and four hours ($\alpha = 0.88$) verifies this. This illustrates the well known fact [3] that the rate at which the load varies affects the actually available charge in the battery. The implication is that for a simple lifetime prediction approach like *Equation (8)*, α may have to be learned on-line if the duty-cycle is not known in advance.

Another complication arises due to the current I_{sys} to compute the lifetime. This is illustrated in the upper graph in *Figure 7*, which shows the runtime predicted by the model. The smooth curve represents the lifetime with a running average for I_{sys}, while the dashed curve represents the prediction when using the instantaneous measured current instead. As is to be expected, in the latter case the prediction follows the voltage profile closely (which follows the drain current). In the former case, the averaging operation smoothens the prediction such that it follows the actual runtime much more closely. Depending on application, either of the approaches may be favorable.

Table 4. Results of the *SOC* approximation performance shown in *Figures 4* – *7*

(a) Cap. error in % (SoC_{model} - SoC_{CC}).

Test	Fig.	Max.	Min.	Mean	Var.	Std.
26 °C	4a	2.98	0.33	2.0	0.44	0.66
−6 °C	4a	3.54	-1.1	0.85	0.96	0.98
Aged	5a	2.41	-4.49	-1.69	3.24	1.8
New	5a	2.04	-5.32	-1.81	5.96	2.44
Var. RL	6	4.98	-0.79	3.02	4.53	2.13
Mean	–	3.19	-2.27	0.47	3.03	1.6

(b) Lifetime error in % of actual runtime.

Test	Fig.	Max.	Min.	Mean	Var.	Std.	α
26 °C	4b	3.85	-1.38	1.4	2.95	1.72	1.0
−6 °C	4b	7.54	-1.34	1.95	2.87	1.7	1.0
Aged	5b	9.21	-3.52	-0.07	9.52	3.09	1.0
New	5b	7.04	-1.13	0.96	4.63	2.15	1.0
Var. RL	7	3.8	-12.29	-0.85	18.83	4.34	0.7
Mean	–	6.29	-3.93	0.68	7.76	2.6	–

Table 4b and the lower graph in *Figure 7* show the prediction error for the running, and absolute average respectively. It would not be fair to compare the instantaneous lifetime prediction to the actual runtime, which is a product of the discharge dynamics. Despite initial large deviations from actual runtime by up to -12.29%, the average error over the entire test is found to be only -0.85%. This is a bit skewed due to consistently, and significantly underestimating during the first half of the test, which exhibits high dynamics in the load variation. However, for the second half, the mean error is roughly 1.9% with a maximum and minimum of 4.52% and 0.04% respectively.

In summary, especially the rate-discharge and to some extent the recovery effects have a large impact on lifetime prediction when load changes occur. As expected, these

effects are less pronounced when a constant load is present. As shown in *Table 4b*, on average the lifetime prediction achieves an accuracy of 98.05% in the worst case (cold battery) and an impressive 99.93% in the best case (aged battery).

Charge Approximation. Approximation of the *SOC* during the charging process works well under optimal conditions. However, when dealing with real-world conditions where the solar panel's performance varies due to *e.g.*, clouds, the state machine introduced in *Section 4.3* has difficulties properly tracking the states, which in turn causes the approximation accuracy to suffer. Considering the relatively good performance during discharge, this is not considered an issue for the moment, but will be further investigated.

6 Conclusions

This paper presented a light-weight approach to battery state-of-charge approximation that relies fully on closed-loop voltage and optional drain current measurements and does not require any special purpose hardware. Instead it leverages the behavior of the battery under load, which is extracted from discharge traces. Therefore, it incurs only limited implementation effort to adapt for a variety of systems. The proposed approach achieves state-of-charge approximation with over 95% accuracy even under varying operating conditions. Minimal operational overhead make this approach suitable for a wide variety of low-performance embedded computing systems that leverage off-the-shelf energy harvesting set-ups.

Acknowledgments. The authors would like to thank their shepherd Leo Selavo for the valuable input. This work was supported by NCCR-MICS, a center funded by the Swiss National Science Foundation under grant number 5005-67322, and the Swiss Nano-Tera.ch initiative.

References

1. Agarwal, V., et al.: Development and Validation of a Battery Model Useful for Discharging and Charging Power Control and Lifetime Estimation. IEEE Transactions on Energy Conversion 25(3), 821–835 (2010)
2. Alessandrelli, D., Azzarà, A., Petracca, M., Nastasi, C., Pagano, P.: ScanTraffic: Smart Camera Network for Traffic Information Collection. In: Picco, G.P., Heinzelman, W. (eds.) EWSN 2012. LNCS, vol. 7158, pp. 196–211. Springer, Heidelberg (2012)
3. Bergveld, H.J.: Battery management systems: design by modelling. Ph.D. dissertation, Enschede (June 2001)
4. Beutel, J., Buchli, B., Ferrari, F., Keller, M., Zimmerling, M., Thiele, L.: X-SENSE: Sensing in extreme environments. In: Design, Automation Test in Europe Conference Exhibition (DATE), pp. 1–6 (March 2011)
5. Buchli, B., et al.: Demo abstract: Feature-rich experimentation for wsn design space exploration. In: Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN 2011), pp. 115–116. ACM/IEEE, Chicago, IL, USA (2011)
6. Buchli, B., Sutton, F., Beutel, J.: GPS-Equipped Wireless Sensor Network Node for High-Accuracy Positioning Applications. In: Picco, G.P., Heinzelman, W. (eds.) EWSN 2012. LNCS, vol. 7158, pp. 179–195. Springer, Heidelberg (2012)

7. Cardei, M., Du, D.-Z.: Improving wireless sensor network lifetime through power aware organization. Wirel. Netw. 11(3), 333–340 (2005)
8. Castagnetti, A., et al.: An efficient state of charge prediction model for solar harvesting wsn platforms. In: 2012 19th International Conference on Systems, Signals and Image Processing (IWSSIP), pp. 122–125 (April 2012)
9. Concorde Battery Corporation, Technical Manual for Lifeline Batteries (July 2011)
10. Doerffel, D., Abu-Sharkh, S.: A critical review of using Peukert-equation for determining the remaining capacity of lead-acid and lithium ion batteries. Journal of Power Sources 155(2), 395–400 (2006)
11. Dunkels, A., et al.: Software-based on-line energy estimation for sensor nodes. In: Proceedings of the 4th Workshop on Embedded Networked Sensors, EmNets 2007, pp. 28–32. ACM, New York (2007)
12. Glatz, P.M., et al.: Designing Perpetual Energy Harvesting Systems explained with RiverMote: A Wireless Sensor Network Platform for River Monitoring. Electronic Journal of Structural Engineering, Special Issue: Wireless Sensor Networks and Practical Applications, 55–65 (2010)
13. Han, J., et al.: State-of-charge estimation of lead-acid batteries using an adaptive extended Kalman filter. Journal of Power Sources 188(2), 606–612 (2009)
14. Hart, J.K., Martinez, K., Ong, R., Riddoch, A., Rose, K., Padhy, P.: An autonomous multisensor subglacial probe: Design and preliminary results from briksdalsbreen, norway. Journal of Glaciology 51(178), 389–397 (2006)
15. James, F.B., Gilbert, M.: Comparison of energy harvesting systems for wireless sensor networks. International Journal of Automation and Computing 5(4), 334 (2008)
16. Jeong, J., Jiang, X.F., Culler, D.E.: Design and Analysis of Micro-Solar Power Systems for Wireless Sensor Networks. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2007-24 (February 2007)
17. Jiang, X., et al.: Micro power meter for energy monitoring of wireless sensor networks at scale. In: Proceedings of the 6th International Conference on Information Processing in Sensor Networks, IPSN 2007, pp. 186–195. ACM, New York (2007)
18. Kerasiotis, F., Prayati, A., Antonopoulos, C., Koulamas, C., Papadopoulos, G.: Battery lifetime prediction model for a wsn platform. In: Proceedings of the 2010 Fourth International Conference on Sensor Technologies and Applications. SENSORCOMM 2010, pp. 525–530. IEEE Computer Society, Washington, DC (2010)
19. Maxim Integrated Products, Evaluation Kit for the MAX1660 (1998), http://datasheets.maxim-ic.com/en/ds/MAX1660EVKIT.pdf (accessed: December 21, 2012)
20. Morningstar Corporation, SunSaver Photovoltaic System Controllers Operator's Manual (November 2009)
21. Park, C., Lahiri, K., Raghunathan, A.: Battery discharge characteristics of wireless sensor nodes: An experimental analysis. In: Proceedings of the IEEE Conf. on Sensor and Ad-hoc Communications and Networks, SECON (2005)
22. Penella Lopez, M.T., et al.: Powering wireless sensor nodes: primary batteries versus energy harvesting (2009)
23. Peukert, W.: Über die Abhängigkeit der Kapazität von der Entladestromstärke bei Bleiakkumulatoren 18, 287–288 (1897)
24. Piller, S., et al.: Methods for state-of-charge determination and their applications. Journal of Power Sources 96(1), 113–120 (2001)
25. Pop, V., et al.: State-of-the-art of battery state-of-charge determination. Measurement Science and Technology 16(12), R93 (2005)

26. Raghunathan, V., et al.: Design considerations for solar energy harvesting wireless embedded systems. In: Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005, pp. 457–462 (April 2005)
27. Randall, J.F.: pp. 1–11. John Wiley & Sons, Ltd. (2006)
28. Rao, R., Vrudhula, S., Rakhmatov, D.: Battery modeling for energy aware system design. Computer 36(12), 77–87 (2003)
29. Taneja, J., Jeong, J., Culler, D.: Design, modeling, and capacity planning for micro-solar power sensor networks. In: International Conference on Information Processing in Sensor Networks, IPSN 2008, pp. 407–418 (April 2008)
30. Vasebi, A., et al.: A novel combined battery model for state-of-charge estimation in lead-acid batteries based on extended Kalman filter for hybrid electric vehicle applications. Journal of Power Sources 174(1), 30–40 (2007); Hybrid Electric Vehicles
31. Weber, S., et al.: Design and measurement assembly to study in-situ rock damage driven by freezing. In: Proceedings of the Tenth International Conference on Permafrost. International Contributions, vol. 1, pp. 437–442. The State Enterprise of the Yamal-Nenets Autonomous District, The Northern Publisher (Severnoye Izdatelstvo), Salekard,Yamal-Nenets autonomous district, Russland (2012)

Author Index