DCOSS ’09
International Conference on Distributed Computing in Sensor Systems
June 8 – 10, 2009
Marina Del Rey, California

Adjunct workshop proceedings
LOCALGOS, IWSNE, WITS
RWSN, DWKDSS, RWI
DCOSS ’09

International Conference on Distributed Computing in Sensor Systems

June 8 – 10, 2009

Marina Del Rey, California

Adjunct workshop proceedings
LOCALGOS, IWSNE, WITS
RWSN, DWKDSS, RWI
Preface

This Proceedings Volume contains the papers of the following Workshops, that were held together with the 5th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '09), which took place at Marina Del Rey, California, USA, during Monday, June 8 - Wednesday, June 10, 2009:

  **Chairs:** WenZhan Song, Washington State University, Vancouver (PC Co-Chair)  
  Hongyi Wu, University of Louisiana at Lafayette (PC Co-Chair)  
  Ivan Stojmenovic, University of Ottawa, Canada (SC Chair)

- Second International Workshop on Sensor Network Engineering (IWSNE)
  **Co-Chairs:** Stefan Fischer, University of Luebeck, Germany  
  Dennis Pfisterer, University of Luebeck, Germany

- Third International Workshop on Information Theory for Sensor Networks (WITS)
  **Chairs:** Joao Barros, Universidade do Porto, Portugal  
  Sandeep Pradhan, University of Michigan, USA

- International Workshop on Robotic Wireless Sensor Networks
  **Chairs:** Andreas Terzis, Johns Hopkins University, USA  
  Volkan Isler, University of Minnesota, USA

- First International Workshop on Data Warehousing and Knowledge Discovery from Sensors and Streams
  **Chair:** Alfredo Cuzzocrea, University of Calabria, Italy

Sotiris Nikoletseas  
DCOSS '09 Workshops Chair
DCOSS Workshop:

Third International Workshop on Localized Algorithms and Protocols for Wireless Sensor Networks (LOCALGOS)
Distributed, Local Energy-Balance Algorithms for Prolonging the Lifetime of Sensor Networks*

Sotiris Nikoletseas
University of Patras &
Research Academic Computer Technology Institute
Greece

Abstract—In this talk, we present recent research on simple, local energy aware mechanisms for prolonging the lifetime of sensor networks. We consider the fundamental problem of data propagation, when sensors collect data from the environment and report it to a particular sink node. Sensors collaborate locally to relay data in a multi-hop fashion towards the sink. However, the geometry of the data-paths from sensors to the sink leads to the overuse of sensors located near the sink and shortens the functioning lifetime of the network by incurring early disconnections although there may be still a lot of energy available in the network as a whole. This simple observation leads to questioning whether balancing the energy consumption among sensors to make their functioning lifetime more or less identical is a relevant strategy. Actually, it turns out that energy-balancing leads to maximizing the network lifetime and thus it is indeed worth designing such energy balance mechanisms.

I. INTRODUCTION

In order to save energy and keep the sensor network functional for as long as possible, various approaches, including hop-by-hop transmission techniques ([6], [2], [3]), as well as clustering techniques ([5]) and alternating power-saving modes ([1]) have been proposed.

All such techniques do not explicitly take care of the possible overuse of certain sensors in the network. As an example, remark that in hop-by-hop transmissions towards the sink, the sensors lying closer to the sink tend to be utilized exhaustively (since all data passes through them). Thus, these sensors may die out very early, thus resulting to network collapse although there may be still significant amounts of energy in the other sensors of the network. Similarly, in clustering techniques the cluster-heads that are located far away with respect to the sink, tend to spend a lot of energy.

*This work has been partially supported by the IST Programme of the European Union under contract number ICT- 2008-224460 (WISEBED).

In this talk, we present protocols trying to balance energy dissipation among the sensors in the network. Basically, we elaborate on the EBP (Energy Balance) protocol, introduced in [4], that probabilistically chooses between either propagating data one hop towards the sink or sending directly to the sink. The first choice is more energy efficient, while the latter bypasses the critical (close to the sink) sectors. The appropriate probability for each choice in order to achieve energy balance is calculated in [4]. Also, we briefly survey some other recent relevant results from the state of the art.

Related Work

An important relevant work is that of [7] where the authors define the energy balance property and propose, analyze and evaluate an energy-optimal and energy balanced algorithm for sorting in wireless sensor networks. In particular, they consider a single-hop sensor network. A similar approach is used in selection. Also, [9] proposes an energy-balanced allocation of a real-time application onto a single-hop cluster of homogeneous sensor nodes connected with multiple wireless channels. An epoch-based application consisting of a set of communicating tasks is considered. Each sensor node is equipped with discrete dynamic voltage scaling (DVS). The time and energy costs of both computation and communication activities are considered. Both an Integer Linear Programming (ILP) formulation and a polynomial time 3-phase heuristic are proposed. Our work extends the approaches above to the general case of a multi-hop network and for the (quite general) problem of propagating data to the sink.

In [10] a new metric, "energy-welfare", is proposed that captures both the average and balance of sensor's remaining energy, and a Maximum Energy Welfare Routing protocol is provided, which achieves energy efficiency and energy balance of sensor networks; the treatment is by simulations and, although good performance gains are achieved, no fundamental relations
between balance and efficiency are provided. In [11] the authors propose a distributed energy balance clustering protocol, in which cluster heads are selected by a probability depending on the ratio between remaining energy of node and the average energy of network. The high initial and remaining energy nodes have more chances to be the cluster heads than the low energy nodes. [12] proposes unequal clustering towards balance the energy dissipation in each cluster; an earlier paper on unequal clustering is [13]. Compared to these approaches, our work is more general (not related to clustering only).

II. EBP: THE ENERGY BALANCE PROTOCOL

A. The Model and the Problem

We assume that crucial events, that should be reported to a control center, occur in the network area. Furthermore, we assume that these events are happening at random uniform positions. Let \( N \) be their total number in a certain period (i.e. during the execution of the protocol).

The sensors are spread in the network area randomly uniformly so their number in a certain area is proportional to the area’s size. There is a single point in the network, which we call the “sink” \( S \), that represents the fixed control center where data about event realization should be reported to. The sink is very powerful, in terms of both energy and computing power. Sensors can be aware of the direction (and position) of the sink, as well as of their distance to the sink. Such information can be easily obtained during a set-up phase, by having the (powerful) sink broadcast control messages to the entire network area. We assume the transmission range of sensors can vary with time (in fact, for each sensor our protocol may use only two different ranges: \( R \) and \( i \cdot R \), where \( i \) is a measure of the sensor’s distance to the sink). The sensors do not move.

We virtually “cover” the network area by a cycle sector of angle \( \phi \) (see Fig. 1). The cycle sector is divided into \( n \) ring sectors or “slices”. The first slice has radius \( R \) (i.e. the sensors’ transmission range). Slice \( i \) (\( 2 \leq i \leq n \)) is defined by two cycle sectors, one of radius \( i \cdot R \) and the other of radius \( (i-1) \cdot R \). Taking a sufficiently large angle \( \phi \) and/or by taking multiple sectors, we can cover the whole area.

As far as energy dissipation is concerned, we assume that the energy spent at a sensor when transmitting data is proportional to the square of the transmitting distance. Our protocol’s performance analysis can be however extended to any energy cost model. Note that the energy dissipation for receiving is not always negligible with respect to the energy when sending such as in case when transmissions are too short and/or radio electronics energy is high. In the analysis, we only count (for simplicity) energy spent during transmissions. Since however in our protocol (see next section) there is one receipt for each transmission, it is clear that even when energy during receipt is more or less the same as energy during transmissions, the analysis can be extended easily to the full case (counting both transmissions and receipts).

Definition 2.1: The area between two consecutive cycle sectors is called a ring sector (or “slice”). Let \( T_i \) (\( 1 \leq i \leq n \)) be the \( i \)-th ring sector of the network. \( T_1 \) stands for the ring sector with center the sink and radius equal to \( R \).

Definition 2.2: Let \( S_i \) be the area size of the ring sector \( T_i \) of the network (\( 1 \leq i \leq n \)).

We wish to solve the “energy balanced data propagation problem”, i.e. to propagate data to the sink in such a way that the “average” energy dissipation in each sensor is at each time the same. The average energy dissipation per sensor is taken to be the fraction of the total energy spent by sensors in a ring sector over the number of sensors in that sector. Because of our assumption that the number of sensors in an area is proportional to the area size, the average energy dissipation per sensor is calculated by dividing the total energy spent in a sector by the sector size.

We do not study here medium access aspects, assuming the existence of underlying MAC protocol.

B. The Protocol

We assume that each event is sensed by only one sensor. This assumption is not restrictive since we may consider multiple sensing and propagation of an event by various sensors as sensing and propagation of many
different events. A sensor sensing an event generates then a data message which should be eventually delivered to the sink. On each ring sector, \( T_i \), a number of events occur and a corresponding number of messages (one for each event) is generated.

Randomization is used to achieve some “load balancing” by evenly spreading the “load” (energy dissipation). In particular, on ring sector \( T_i \) each event is propagated to \( T_{i-1} \) (i.e. the “next” sector towards the sink) with probability \( p_i \), while with probability \( 1 - p_i \) it is propagated directly to the sink \( S \). Each message in \( T_i \) is handled stochastically independently of the other events’ messages.

The choice of probability \( p_i \) for \( T_i \) is made so as the average energy consumption per area unit (and thus per sensor) is the same for the whole network. There is a trade-off from choosing \( p_i \): if \( p_i \) increases then transmissions tend to happen locally, thus energy consumption is low, however sensors closer to the sink tend to be overused since all data passes through them. On the other hand, if \( p_i \) decreases, there are distant transmissions (thus a lot of energy is consumed) however closer to sink particles are bypassed. Calculating the appropriate probability \( p_i \) for each \( T_i \) and solving the problem of energy balance is very important since it combines efficient data propagation with increased network’s lifetime.

By using an underlying subprotocol ([3], [6]) we can guarantee that only one “next hop” sensor receives the transmitted message. Note also that data messages are of fixed size i.e. no further info is added to a message on its route towards the sink.

Our protocol is (a) distributed, since each sensor chooses propagation probability independently of other sensors, (b) it uses only local information, in the sense that \( p_i \) depends only on \( i \), i.e. a parameter related to the distance from the sink. Note that distance from the sink info for each sensor can be easily obtained i.e. during a set-up phase where the sink broadcasts control messages to the network. Several techniques (including signal attenuation evaluation) can be used to estimate each sensor’s distance from the sink. (c) The protocol is very simple, since it just uses a random choice based only on parameter \( i \).

**C. Basic Definitions - Preliminaries**

We aim at calculating probability \( p_i \) for each \( i \) in order to ensure the energy balance property. Using simple geometry, one can easily prove the following Lemmas.

**Lemma 2.1**: The area size, \( S_1 \), of the ring sector \( T_1 \) is \( S_1 = \frac{\pi}{2} \cdot R^2 \).

**Lemma 2.2**: The relation between the area size of the ring sector \( T_i \) and that of \( T_1 \) is \( S_i = (2i - 1) \cdot S_1 \).

**Definition 2.3**: Let \( \lambda_i \) the probability that an event will occur on the ring sector \( T_i \).

There are \( n \) ring sectors in the network.

**Lemma 2.3**: Assuming a random uniform generation of events in the network area, the probability \( \lambda_i \) of an event occurring on the ring sector \( T_i \) (\( 1 \leq i \leq n \)), is:

\[
\lambda_i = \frac{(2i - 1)}{n^2}
\]

Let us now consider sector \( T_i \).

**Definition 2.4**: An area \( T_i \) “handles” an event generated in ring sector \( j \) if either the message was generated in the area \( T_i \) (i.e. \( j = i \)) or the message was propagated to \( T_i \) from the ring sector \( T_{i+1} \).

**Definition 2.5**: Let \( h_i \) be the number of the messages that are “handled” by the area \( T_i \).

We now define energy \( \epsilon_{ij} \) spent for message \( j \) when sector \( i \) handles it.

**Definition 2.6**: Let \( \epsilon_{ij} \) a random variable which measures the energy that dissipates the sector \( T_i \) so as to handle the message \( j \). For \( \epsilon_{ij} \) we have that:

\[
\epsilon_{ij} = \begin{cases} 
  cR^2 & \text{with probability } p_i \\
  c(iR)^2 & \text{with probability } 1 - p_i
\end{cases}
\]

where \( cR^2 \) is the energy dissipation for sending a message \( j \) from \( T_i \) to its adjacent ring sector \( T_{i-1} \) and \( c \) is a constant.

Thus, the expected energy dissipation in sector \( i \) for handling a message is

\[
E[\epsilon_{ij}] = cR^2 \cdot [i^2 - p_i(i^2 - 1)]
\]

**Note**: The expected energy above is the same for all messages; we use \( j \) just for counting purposes.

**Definition 2.7**: Let \( E_i \) the total energy spent by sensors in \( T_i \). Clearly:

\[
E_i = \sum_{j=1}^{h_i} \epsilon_{ij}
\]

Energy balance is defined as follows:

**Definition 2.8**: The network is energy balanced if the average per sensor energy dissipation is the same for all sectors, i.e. when

\[
\frac{E[E_i]}{S_i} = \frac{E[E_j]}{S_j} \quad i, j = 1, \ldots, n
\]
D. The General Solution

We next provide a lemma useful in the estimation of the total energy dissipation in a sector.

**Lemma 2.4:** The expected total energy dissipation in sector i is:

\[ E[E_i] = E[h_i] \cdot E[\epsilon_{ik}] \]

**Proof:**

\[ E[E_i] = E \left[ \sum_{k=1}^{N} \epsilon_{ik} \right] = \sum_{n=0}^{N} E \left[ \sum_{k=1}^{N} (\epsilon_{ik} \cap h_1 = n) \right] = \sum_{n=0}^{N} E \left[ \sum_{k=1}^{N} \epsilon_{ik} \right] \cdot P\{h_i = n\} \]

Furthermore,

\[ E \left[ \sum_{k=1}^{N} \epsilon_{ik} \right] \cdot h_1 = n = E \left[ \sum_{k=1}^{N} \epsilon_{ik} \right] \]

Thus, we have from the above that:

\[ E[E_i] = \sum_{n=0}^{N} E \left[ \sum_{k=1}^{N} \epsilon_{ik} \right] \cdot P\{h_i = n\} = E[\epsilon_{ik}] \cdot E[h_1] \]

**Definition 2.9:** Let \( g_i \) be the number of the messages that are generated in the area \( T_i \).

Note that messages are generated in an area only when events occur in this area.

**Definition 2.10:** Let \( f_i \) be the number of the messages that are forwarded to the area \( T_i \).

We note that messages are forwarded to a ring sector (say \( i \)) only because of an event generation at a sector \( j > i \) and successive one-hop propagations from sector \( j \) to sector \( i \).

We notice the following important relation:

\[ h_i = g_i + f_i \]  \hspace{1cm} (4)

which means that the number of messages that area \( T_i \) handles equals the number of the messages that are generated in \( T_i \), plus the number of messages that are forwarded to it. Because of event generation according to a probability distribution and also because of the probabilistic nature of message propagation in, all three quantities above are random variables. By linearity of expectation, we get:

**Lemma 2.5:** \( E[h_i] = E[g_i] + E[f_i] \)

We establish a relationship between \( E[f_i] \) and \( E[h_{i+1}] \).

**Lemma 2.6:** \( E[f_i] = p_{i+1} \cdot E[h_{i+1}] \)

**Proof:** Let \( \delta_{i,j} \) an indicator random variable that is equal to 1 if area \( T_i \) forwards the message \( j \) to the area \( T_{i-1} \) and 0 otherwise. Thus:

\[ \delta_{i,j} = \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{with probability } 1 - p_i \end{cases} \]

Clearly, \( \delta_{i,j} \) depends only on \( i \), but we add \( j \) for counting purposes. Obviously, \( E[\delta_{i,j}] = p_i \). It is:

\[ f_i = \sum_{j=0}^{h_{i+1}} \delta_{i+1,j} \]

Similarly to the proof of Lemma 2.4, we get:

\[ E[f_i] = \sum_{n=0}^{N} E \left[ \sum_{j=0}^{h_{i+1}} \delta_{i+1,j} \right] \cdot P\{h_{i+1} = n\} \]

and the proof is completed.

Recall that, according to Definition 2.8, to achieve the same on the average energy dissipation per area unit (and thus per sensor) in the network area, the following equality should hold:

\[ \frac{\sum_{k=1}^{N} \epsilon_{ik}}{S_i} = \frac{\sum_{k=1}^{N} \epsilon_{jk}}{S_j} \quad \forall i, j \in \{1, \ldots, n\} \]  \hspace{1cm} (5)

i.e. the average energy consumption per sensor should be equal in any two ring sectors. By induction, it suffices to guarantee this for any two adjacent sectors. In what follows, we guarantee the above balance property, requiring a certain recurrence relation to hold. This recurrence basically relates 3 successive terms of the \( E[f_i] \) sequence (the \( E[g_i] \) terms depend only on \( i \) and on input parameters).

**Theorem 2.7:** To achieve energy balance in the network, the following recurrency equation should hold:

\[ a_{i+1}E[f_{i+1}] - (d_i + a_i)E[f_i] + d_{i-1}E[f_{i-1}] = a_iE[g_i] - a_{i+1}E[f_{i+1}] \]

where

\[ a_i = \frac{j^2}{2i-1} \quad d_i = \frac{(i+1)^2-1}{2i+1} \]

**Proof:** For the case \( j = i + 1 \) of the equation 5 and using the lemmata 2.2 and 2.4 we have:

\[ E[h_i] = E[\epsilon_{i,j}] = E[h_{i+1}]E[\epsilon_{i,j}] \frac{S_i}{S_{i+1}} \Leftrightarrow \]

\[ E[h_i] \left[ \frac{(i+1)^2-1}{2i+1} \right] = E[h_{i+1}] \left[ \frac{(i+1)^2-1}{2i+1} \right] \Leftrightarrow \]

\[ \frac{j^2}{2i-1} E[h_i] = p_i E[h_i] \frac{(i+1)^2-1}{2i+1} \]

\[ = \frac{(i+1)^2}{2i+1} E[h_{i+1}] - p_{i+1} E[h_{i+1}] \frac{(i+1)^2-1}{2i+1} \]

Let \( a_i, d_i \) as defined in the Theorem statement above. By lemma 2.6 we know that \( p_i E[h_i] = E[f_{i-1}] \) and
by lemma 2.5 it is \( E[h_i] = E[g_i] + E[f_i] \) thus the last equation becomes:

\[
a_{i+1}E[f_{i+1}] - (d_i + a_i)E[f_i] + d_{i-1}E[f_{i-1}] = \\
= a_iE[g_i] - a_{i+1}E[g_{i+1}]
\]

To solve the above recurrence we must compute \( E[g_i] \).

**Lemma 2.8:** If \( N \) is the total number of events that are generated in the network, the mean value of \( g_i \) is given by the following relationship:

\[
E[g_i] = N \cdot \lambda_i
\]

**Proof:** Because the position of each event is independent of other events and because for each sector \( i \), the probability \( \lambda_i \) is the same, clearly \( g_i \) is binomial with parameters \( N \), \( \lambda_i \).

In order to have a simpler recurrence involving only two (successive in fact) terms of the \( E[f_i] \) sequence, we will transform the recurrence relation of Theorem 2.7 into the following (easier to solve) relation:

**Lemma 2.9:** The recurrence relation:

\[
t_i - t_{i-1} = a_i \cdot E[f_i] - a_{i+1} \cdot E[f_{i+1}]
\]

for \( i = 1, \ldots, n - 1 \) and \( t_0 = a_1 \cdot E[f_1] \)

has as a solution the function

\[
t_i = \sum_{j=1}^{i} (a_jE[g_j] - a_{j+1}E[g_{j+1}]) + a_1 \cdot E[f_1]
\]

**Proof:** The proof is done by induction on \( i \). For \( i = 0 \), it is obviously true. Let it be true for \( i - 1 \). For \( i \) we have:

\[
t_i = t_{i-1} + a_i \cdot E[g_i] - a_{i+1} \cdot E[g_{i+1}]
\]

By the induction hypothesis we get the solution

\[
t_i = \sum_{j=1}^{i} (a_jE[g_j] - a_{j+1}E[g_{j+1}]) + a_1 \cdot E[f_1]
\]

Now the recurrence relation of Thrm 2.7 is simplified:

\[
a_{i+1} \cdot E[f_{i+1}] - d_i \cdot E[f_i] = t_i \quad i = 1, \ldots, n - 1
\]

Thus, we get a recurrence for sequence \( E[f_i] \) involving only two successive terms of the sequence:

**Theorem 2.10:** The recurrence relation

\[
a_{i+1}E[f_{i+1}] - d_iE[f_i] = t_i \quad i = 1, \ldots, n - 1
\]

where \( t_i \) is defined in lemma 2.9, has the following solution

\[
E[f_{n-i}] = -\sum_{k=1}^{i} \frac{\prod_{j=k}^{n-1} a_{n-j}}{\prod_{j=k}^{n-1} d_{n-j}} \cdot t_{n-k}
\]

The proof is rather complex, so we omit it here. The interested reader may find it in [13].

The full expression for \( E[f_i] \) can be expressed by substituting \( i \) with \( n - i \), thus

\[
E[f_i] = -\sum_{k=1}^{n-i} \frac{\prod_{j=k}^{n-i} a_{n-j}}{\prod_{j=k}^{n-i} d_{n-j}} \cdot \left( \sum_{j=1}^{n-k} (a_jE[g_j] - a_{j+1}E[g_{j+1}]) + a_1 \cdot E[f_1] \right)
\]

where

\[
\prod_{j=1}^{n-i} a_j = 1.
\]

We note that all the parameters of the recurrence solution above are expressed as a function of \( E[f_1] \) and \( i \). So as to compute them, we firstly compute the value of \( E[f_1] \). Then we can compute all the other parameters by replacing the already computed \( E[f_1] \).

Now, the calculation of the probabilities \( p_i \) is quite easy.

**Theorem 2.11:** The energy balance property is achieved if any ring sector (say \( T_i \)) propagates each message it handles with probability \( p_i \) to the next ring sector, \( T_{i-1} \), and with probability \( 1 - p_i \) it propagates the message directly to the sink. The value of each \( p_i \) is given by the following relation

\[
p_i = \frac{E[f_{i-1}]}{E[g_i] + E[f_i]}
\]

where the values of \( E[f_1] \) and \( E[g_i] \) are obtained from lemma 2.10 and lemma 2.8, respectively.

**Proof:** From equation 2.6 we know that

\[
E[f_{i-1}] = p_iE[h_i] \quad \text{and also by lemma 2.5 we know that} \quad E[h_i] = E[g_i] + E[f_i].
\]

**Remark.** Note that, interestingly, \( p_i \)'s are independent of the number \( N \) of the events that occur in the network, since \( p_i \)'s depend only on \( i \) and the number of ring sectors \( n \) (which is broadcast to sectors by the sink). Thus the protocol assumes only local information.

We note that the analysis above allows the exact derivation of probabilities \( p_i \)'s as a function of \( i \) and \( n \) which (although complicated and not obviously leading to a closed form) can be easily calculated by the sensors in the network by carrying out very simple calculations.

The authors of [4] also prove the correctness of the protocol:

**Theorem 2.12:** Given that the energy is each time on the average the same in all network sensors, each message will finally get to the sink.

**E. A Closed Form**

Under specific assumptions (that we discuss and motivate) we can make the calculation of probabilities.
\( p_i \) simpler. Combining lemma 2.5 and lemma 2.8 we have that
\[
E[h_i] = \lambda_i N + E[f_i] = \frac{2i-1}{n^2} N + E[f_i] \quad (6)
\]
By the corresponding relation for \( E[h_{i-1}] \) it must be:
\[
\frac{2i-2N+E[f_i]}{(2i-1)^2} R^2 = \frac{2i-2N+E[f_{i-1}]}{(2i-3)^2} R^2 = \frac{p_i + (1-p_i)^2}{p_{i-1} + (1-p_{i-1})(i-1)^2} R^2
\]
But \( 2i - 1 \approx 2i - 3 \) and \( \phi \), \( R^2 \) cancel. Dividing by \( N \) we get:
\[
\left[ 1 + \frac{E[f_i]}{N} \right] (p_i + (1-p_i)i^2) \approx \left[ 1 + \frac{E[f_{i-1}]}{N} \right] (p_{i-1} + (1-p_{i-1})(i-1)^2)
\]
If \( E[f_i] \approx E[f_{i-1}] \) then the previous relation becomes:
\[
p_i + (1-p_i)i^2 = p_{i-1} + (1-p_{i-1})(i-1)^2 \quad (7)
\]
In [4] we show how to solve the above recurrence.

Theorem 2.13: If \( E[f_i] \approx E[f_{i-1}] \), \( 3 \leq i \leq n \), then the one-hop forwarding probability, guaranteeing energy balance, is
\[
p_i = 1 - \frac{3x}{(i+1)(i-1)}
\]
where \( p_2 = x \in (0, 1) \) a free parameter and \( p_1 = 0 \).

We remark that the assumption \( E[f_i] \approx E[f_{i-1}] \) is quite reasonable and well motivated. We provide the following intuitive explanation of why this happens. Remark indeed that the area sizes of adjacent sectors (and thus the number of events generated in such sections) are more or less the same, especially when \( i \) increases. Furthermore, the probability \( p_i \) of forwarding to the adjacent (towards the sink) sector increases very fast with \( i \) and becomes 1 in most sectors of the network (in the middle territory).

III. RECENT RESULTS

In [16] and [15] the authors consider the problem of data propagation in wireless sensor networks and revisit the family of mixed strategy routing schemes. They show that maximizing the lifespan, balancing the energy among individual sensors and maximizing the message flow in the network are equivalent. They propose a distributed and adaptive data propagation algorithm for balancing the energy among sensors in the network. The mixed routing algorithm they propose allows each sensor node to either send a message to one of its immediate neighbors, or to send it directly to the base station, the decision being based on a potential function depending on its remaining energy. By considering a simple model of the network and using a linear programming description of the message flow, they prove the strong result that an energy-balanced mixed strategy beats every other possible routing strategy in terms of lifespan maximization. Moreover, they provide sufficient conditions for ensuring the dynamic stability of the algorithm. The algorithm is inspired by the gradient-based routing scheme but by allowing to send messages directly to the base station they improve considerably the lifespan of the network. As a matter of fact, they show experimentally that their algorithm is close to optimal and that it even beats the best centralized multi-hop routing strategy.

REFERENCES

A Localized Routing Metric for Solar-powered Wireless Sensor Networks

Jing Hu, Lianfeng Shen and Yanli Xu
National Mobile Communications Research Laboratory
Southeast University
Nanjing, China
e-mail: [louy, lfshen]@seu.edu.cn, xylzoe1@163.com

Abstract—Rapid advances in wireless sensor networks require routing protocols to accommodate new types of power source. And recent experimental studies have shown that in real (realistic) sensor network deployments, wireless link quality varies over space and time, deviating from the idealized unit disc graph models used in network simulation tools. In this paper, we propose a routing metric combining residual energy, link quality and usage probability (ELP) for solar-powered wireless sensor networks connected as grids, which can be only calculated with localized data collected from adjacent nodes without the global information of the network. Performance evaluations show that the ELP metric achieves high performance in terms of packet reception rate, throughput and lifetime.

Keywords- Wireless Sensor Networks, link quality, routing metric

I. INTRODUCTION

A wireless sensor network (WSN) is a network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions. And now, wireless sensor networks have become a rather active research area. Recent advances in wireless sensor networks have led to many new protocols specifically designed for sensor networks where energy awareness is an essential consideration. Most of the attention has been paid to the routing protocols since they might differ depending on the network architecture [1].

Precision agriculture (PA) is a research discipline based on the investigation of the differences that exist in every cultivated field. By exploiting such in-field variability, it is possible to manage crop irrigation or fertilization, etc. on a site-specific basis to increase profit, reduce waste and maintain environmental quality. Wireless sensor networks are a new technology that can provide viable solutions for PA applications [2]. A wireless sensor network may consist of sensor and actuators nodes, one sink node or gateway to connect WSN with other wide-area networks such as cellular networks and Internet, and software for monitoring and controlling the whole system. The sensor and actuators nodes developed up till now are intended for variable rates micro drip irrigation based on soil parameters, and they are usually line or solar powered to extend the node lifetime. Routing protocols must be developed to manage the data acquisition and control actions of the electromagnetic valves. In this paper, we present a localized routing metric for Solar-powered Wireless Sensor Networks based on energy resource and link quality.

The rest of this paper is organized as follows. The next section contains an analysis of existing routing protocols. Sections 3 and 4 describe the characteristics of ELP, our new routing metric based on a solar-cell energy model. We then give an overview of our simulation environment and evaluate our strategy in Section 5. Finally, conclusions are drawn and future work is discussed in Section 6.

II. RELATED WORKS

One of the main design goals of routing protocols for WSN is to minimize the energy expenditure of individual nodes or balance the energy expenditure between neighboring nodes so as to extend the lifetime of the whole networks, such as LEACH[3], PEGASIS[4], etc. Most routing protocols take into account distances between nodes and energy efficiency. But recent experimental studies have shown that in practical sensor network deployments, wireless link quality varies over space and time, deviating to a large extent from the idealized unit disc graph models used in network simulation tools. Packet losses due to fading and obstacles are common at a wide range of distances and keep varying over time. So only shortest path is not enough [5]. On the other hand, the design space of sensor platforms and their radio hardware have advanced significantly. Recently, many sensor platforms have gravitated towards an international sensor network standard (IEEE 802.15.4 [6]) which provides an additional radio hardware link quality indicator (LQI). Consequently, these new advances in sensor radio hardware as well as link quality variations and node heterogeneities in WSNs call for new designs of resource-aware protocols for WSNs.

In the literature, most studies of energy-aware routing focus on the residual battery status and do not take into account the environmental energy availability at the nodes. The first approach to utilization of environmental energy for routing [7] demonstrated that environmentally aware decisions improve performance compared to decisions based only on battery status. A solar-aware version of directed diffusion has also been proposed in [8]. Like directed diffusion, this data propagation protocol is based on localized interactions (i.e.
interactions between neighboring nodes). Gradients are used to provide additional information about the state of neighbors, which may be running on solar power, or relying on their batteries.

Sensor placement has been an active research topic recently. All of them address the environment where the initial deployment is random and later the sensors will spread themselves through the environment. But in applications where deliberate deployment is possible and sensors are static, such as those in healthcare applications, traffic control systems, gallery monitoring systems, etc., deploying sensors in a regular topology like a grid can make the network more resilient and energy efficient. Grid topology has been studied by network researchers for many years. It started as a simple topology with good properties for interconnection networks. The related researches were focusing on finding an optimal routing path that maximizes the probability of reaching the destination from a given source [9]. Later when ad hoc wireless networks appeared, grid was proved to be a better topology than a tree due to its many connections and redundant paths [10]. [11] discuss energy balancing routing topology than a tree due to its many connections and redundant paths [10]. [11] discuss energy balancing routing.

The energy got during the period $T$ can also be presented by the equation as follows:

$$require(T) = \int_{t}^{T} P_{\text{solar}}(x)dx$$

where $P_{\text{solar}}(x)$ is the energy from solar radiation per unit time.

To transmit and receive a $k$-bit message over a distance $d$, the radio expends

$$E_{\text{Fs}}(k) = kE_{\text{elec}}$$

where $E_{\text{elec}}$ is the energy required to run the transmitter or receiver circuitry with a typical value of 50 nJ/bit, $d_0$ is the cross-over distance with a typical value of 86.2 m, $\epsilon_{\text{fs}}$ and $\epsilon_{\text{amp}}$ is the energy needed for the transmitter amplifier when $d$ is less or more than $d_0$ and has the typical value of 10 pJ/bit/m² and 0.0013 pJ/bit/m⁴ [3].

### III. PRELIMINARY

#### A. Network Model

We define a wireless sensor network as a graph $G = (V, E)$, where $V$ is the set of all sensors and $E$ is the set of all edges between pairs of sensors. If two sensors are within each other’s transmission range, there is an edge between them in $G$. In this paper, we assume that all sensors are deployed in a grid topology. The location of each sensor is represented by $(x, y)$. Without loss of generality, we assume $x \geq 0$, $y \geq 0$, and the sink is treated as a special sensor of the location (0,0). If node $(i, j)$ is not on the edge of the grid, it has four neighbors, namely $(i+1, j)$, $(i-1, j)$, $(i, j+1)$, and $(i, j-1)$. For simplicity, we can just discuss the communication between $(i, j)$ and $(i+1, j)$, $(i-1, j)$, $(i, j+1)$, and $(i, j-1)$. Figure 1 shows 16 sensors are deployed in a $4 \times 4$ grid, where the sink $(0, 0)$ is located at the left down corner.

#### B. Energy Model

We assume the nodes are enhanced with the solar rechargeable batteries. The battery update scheme is given by the equation as follows:

$$energy^t = energy^{t-1} - consume^{t-1} + require^{t-1}$$

where $energy^t$ and $energy^{t-1}$ are the battery capacities of a sensor at the beginning of measuring cycles $t$ and $t-1$ respectively; $consume^{t-1}$ is the battery reducing rate factor for the sensor during measuring cycle $t-1$; $require^{t-1}$ is the amount of battery capacity added to each sensor from solar radiation available during measuring cycle $t-1$.

Figure 1. A $4 \times 4$ sensor network connected as a grid
possible. Take a look at Figure 1, and suppose the source is (0, 0) and the destination is (3, 2). If we assume only two flow directions are allowed, that’s to say, rightwards and upwards, there will be 5 hops between the source and the destination. Because the 5 hops must be composed of 3 rightwards and 2 upwards segments, there are 10 kinds of permutations totally, which correspond to 10 different paths. If the probability that each path will be used is the same, we find that the number of times each sensor is used which is called sensor usage is different. For example, in the 10 paths Sensor (1, 2) and (2, 0) are used 3 times, while sensor (2, 1), (2, 2) and (1,1) are used 6 times. So we can find that if a packet is randomly routed using any of these paths from a source to a destination, some sensors will be used more often than others and thus their energy will be depleted more quickly.

Generally in a sensor network connected as an m \times n grid, if the source is (0, 0) and the destination is (u, v), the total number of paths between them is \(\binom{u + v}{v}\), and the probability that intermediate sensor \((i, j)\) \((0 \leq i \leq u, 0 \leq j \leq v)\) will be chosen is

\[
proportion = \frac{(j + i)}{j} \times \frac{(u + v - j - i)}{v - j} \quad (5)
\]

which is called the usage probability of a sensor. Suppose its two neighbors’ (sensors \((i, j + 1)\) and \((i + 1, j)\)) usage probabilities are \(p_1\) and \(p_2\), it should generate a random number such that the probability that sensor \((i, j + 1)\) is chosen is \(1/p_1\),and the probability that sensor \((i + 1, j)\) is chosen is \(1/p_2\).

C. ELP

In the above two algorithms, routing metrics are based on stable information such as locations of the nodes. Although The Proportional Algorithm tries to balance the usage of the sensors, it does not consider the residual energy of the nodes and link quality. In the non-ideal environment, even though nodes have almost the same usage probability, worse link quality will cause much more energy expenditure, so some nodes will deplete their residual energy more quickly.

So we try to select an appropriate metric according to energy efficiency, link quality and usage probability. We use a cost function \(f\) to represent the weight value of the next hop. Sensor nodes will select the path with higher weight value to relay the packet. The cost function can be calculated as follows:

\[
f = \alpha \times \text{energy} + \frac{\beta}{Q_{\text{link}}} + \frac{r}{\text{proportion}} \quad (6)
\]

where the variable \(\alpha, \beta\) and \(\gamma\) are weighting factors that can be adjusted; energy represents the remaining energy of next node; \(Q_{\text{link}}\) represents the expected number of transmissions;

\[
\begin{align*}
Q_{\text{link}} & = \gamma (1 - \text{PRR})^{K} \text{PRR} \quad (7)
\end{align*}
\]

Where PRR represents the packet reception rate and \(K\) is the maximum number of re-transmissions before the packet is ignored. To calculate PRR of a link, we utilize link quality indicator (LQI) reported by the physical layer of IEEE 802.15.4 [6]. In this way, the nodes dynamically adapt to changing wireless network conditions and select the paths with high quality links.

V. SIMULATION RESULTS AND PERFORMANCE ANALYSIS

In this section, we evaluate the performances of ELP obtained by simulation utilizing OPNET. A 4 \times 4 sensor network connected as a grid is generated, the same as figure 1, with the space of 20m between a pair of nodes. A sensor’s battery is charged by an amount of 2% of initial battery energy during the simulation time. Three links are added noise randomly. Other parameters used in the simulation are listed in table 1.

In the evaluations, we investigate the following performance metrics:

- **Throughput** is the number of unique packets received at the sink node divided by the interval between the start and the end of the experiment.
- **Packet Reception Rate** is the ratio between the total number of unique packets received at the sink node and the total number of packets generated by all the sensor nodes.
- **Left nodes** are defined as the number of nodes which have not depleted energy. To speed up the simulation, we let \(\text{require}\) in equation 1 to be zero, which can be considered as the scene when terrible weather lasts for a long period.

<table>
<thead>
<tr>
<th>Parameter Names</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area of sensor field</td>
<td>100×100 m²</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>16</td>
</tr>
<tr>
<td>Packet length</td>
<td>32 bytes</td>
</tr>
<tr>
<td>Transmission range</td>
<td>25 m</td>
</tr>
<tr>
<td>Transmission power</td>
<td>-20dbm</td>
</tr>
<tr>
<td>Packet generate rate</td>
<td>1 packet/second</td>
</tr>
<tr>
<td>Weighting factors((\alpha, \beta, \gamma))</td>
<td>(5,1,0.1)</td>
</tr>
<tr>
<td>Initial energy</td>
<td>0.125 J</td>
</tr>
<tr>
<td>Simulation time</td>
<td>1500 s</td>
</tr>
</tbody>
</table>
Figure 2, 3 and 4 plot the simulation results of the three metrics above. Figure 2 and 3 show that ELP has better throughput and PRR, because ELP intend to choose paths with better link quality, thus reduces the number of dropped packets. Figure 4 shows that network has longer lifetime using ELP. If the residual energy of a node becomes lower, then the residual energy term is more emphasized. With the use of proposed metric, we can choose paths that contain as more residual energy data transmissions and receptions as possible and thus utilizes resource-rich nodes in the deployment field in order to maximize the network lifetime.

Then we study the different choices of the three weight factors in equation 6. Firstly, we let $(\alpha,\beta,\gamma) = (0,1,0.1)$, that’s to say, do not consider the residual energy. Simulation results are presented in figure 5. It shows that residual energy plays a rather important role to extend the network lifetime, because some nodes may be overburden relay traffic and die more quickly.

Next we let $(\alpha,\beta,\gamma) = (5,0,0.1)$, while omit the link quality. We find that the link quality impacts throughput, because bad link radio will lead packet loss after maximum number of re-transmissions. Figure 6 shows the simulation result.
VI. CONCLUSION

In this paper, we have put forward a localized routing metric ELP for Solar-powered Wireless Sensor Networks, which is based on residual energy, link quality statistics and usage probability. Besides the solar-cell energy model, the results in this paper can also be applied to line or battery powered nodes. If the failure of some sensors blocks all the shortest paths between a source and a destination, then detours will be considered. And besides grid, other topologies should be explored. These will be our considerations in the future work.

REFERENCES

A Distributed Location Management Scheme for 3D Container Sensor Networks

X. Yu, S. Kim, V. Sarangan, X. Li
Computer Science Department
Oklahoma State University
Stillwater, OK 74078, USA
Email: {xinjie, sinil, saranga, xiaolin}@cs.okstate.edu

S. Radhakrishnan
School of Computer Science
University of Oklahoma
Norman, OK 73019, USA
Email: sridhar@ou.edu

T. Landers
School of Industrial Engineering
University of Oklahoma
Norman, OK 73019, USA
Email: landers@ou.edu

Abstract—Localization has been one of the central issues and key challenges in wireless sensor networks. However, most of the current localization research effort is devoted to two dimensional (2D) sensor network applications through range based or range free techniques. Motivated by the real world application of round-the-clock monitoring of freight containers, we propose a distributed location management scheme for 3D container sensor networks. The proposed scheme takes advantage of the underlying container network's structure and the relative differences in the received signal strength between different network nodes to discover each node's location. A distinguishing feature of the proposed work is that, unlike most of the received signal strength indicator (RSSI) based algorithms, our algorithm is not required to do any calculation based on the absolute value of RSSI. Therefore, our algorithm is able to avoid the notorious RSSI accuracy issue caused by such phenomena as shadowing and path loss. Performance studies undertaken using a rudimentary 3D container sensor network simulator show that the proposed localization scheme works satisfactorily and has acceptable time and message complexities.

I. INTRODUCTION

Securing cargo movement through the international supply chains that service American consumers and businesses is critical to the security of national infrastructure. Central to this motive is the subsistence of a safe, efficient, and uninterrupted operation of the nation’s transportation modes, ports, and inter-modal terminals, which in turn, presumes effective examination and tracking of container contents. The United States ranks second in the world in terms of maritime container traffic with over 46 million 20-foot equivalent units of container passing through the US ports in 2006 [13]. With such an enormous volume of traffic, manual inspection of all the containers is impossible and even if achieved, the delay involved can be extremely detrimental to the supply chain economics. In addition, the bogey of delivering chemical, biological, radiological and/or nuclear weapons through containers has also motivated the need for reliable, cost-effective technologies that can automate container monitoring while guaranteeing operational efficiency at the nation’s ports.

Though some commercial container monitoring solutions exist [1], [24], they suffer from one or more of the following drawbacks: (i) the querying range is extremely limited (not more than few meters) and hence a container cannot be monitored if it is not in the immediate vicinity of the querying entity (such as the customs personnel); this in turn implies that huge amounts of resources have to be invested, in terms of number of querying equipment and/or man power to move the containers close to the querying equipment to achieve a high degree of monitoring. (ii) the solutions may also demand that each container be installed with GPS tracker and satellite communication links which can be very expensive to operate, both in terms of cost and battery power consumption.

As wireless sensor networks have been regularly used for environmental monitoring and surveillance applications, they present themselves as natural candidates for container monitoring. Each container could be assigned an unique identifier and fitted with a wireless sensing device. Through the sensing device the status of a container can be remotely and automatically monitored. Whenever a suspicious activity is detected in the vicinity of a container, appropriate human personnel can be sent to do a physical inspection. At the heart of implementing such a round-the-clock container monitoring system is the ability to discover the location of a specific container located in a port yard or a ship hull, where it could be stored along with thousands of other containers. This gives rise to the location management problem in real-world container networks.

Note that in container sensor networks, containers are typically stacked on top of one another. Thus, unlike most other sensor network applications, container sensor networks are three dimensional and so is the location management problem. As will be clear later in section II, though there are few 3D location management schemes have been proposed for sensor/RFID networks, they may not readily applicable to the container sensor networks.

In this paper, we propose a distributed location management scheme for 3D container sensor networks. The proposed scheme leverages on the underlying container network’s structure and the relative differences in the received signal strength between different network nodes. A distinguishing feature of the proposed work is that unlike most of the received signal strength indicator (RSSI) based algorithms, our algorithm is not required to use the absolute value of RSSI and thus can be more robust to shadowing and path loss variations. Performance studies on a rudimentary 3D container sensor
network simulator show that the proposed localization scheme works satisfactorily and has acceptable time and message complexities.

The rest of this paper is organized as follows. In Section II, we discuss the research efforts related to the proposed work. In Section III, we describe the proposed 3D location management scheme. In Section IV, we highlight the performance of the proposed scheme through simulation studies. We finally conclude in Section V with a brief discussion of future work.

II. RELATED WORK

There has been considerable amount of work done in the area of localization in 2D sensor networks. Existing approaches in 2D localization can be generally divided into range-free and range based methods. Stoleru et al [19] gave a detailed survey of the range-free localization solutions, which is further divided into anchor-based solutions and anchor free solutions. Centroid in Bulusu et al [4] (the idea is to treat the anchor nodes as point of masses and to find the center of gravity), area-based scheme in He et al [7] and Lazos et al [9], MultiDimensional Scaling (MDS) data analysis technique in Shang et al. [18], Monte Carlo based method in Rudafshani et al [16], Gradient estimation scheme in Nagpal et al [10], Ad-hoc (DV-hop) Positioning System in Niculescu et al [11] and Probability Grid in Stoleru et al [20] are all belong to anchor-based solutions. Meanwhile, anchor free based solution take advantage of proximity to an event with a known location, such as a light event in Stoleru et al [22] and Romer et al [15] or a nearby radio packets in Stoleru et al [21]. On the other hand, range based methods use range measurements, such as Received Signal Strength (RSS) in Bahl et al [2] and Bischoff et al [3], Time of Arrival in Ward et al [26]or Time Difference of Arrival in Priyantha et al [14] and Savvides et al [17], Angle of Arrival in Niculescu et al [12], and discrete position distribution in the Dil et al [6] Sequential Monte Carlo Localization solution.

In the area of 3D localization in sensor networks, there are very few works. Zhang et al [28] proposed a Landscape-3D scheme by introducing a location assistant (such as GPS or predefined moving path) to the sensor networks and then solve the functional dual tracking problem by an Unscented Kalman Filter based algorithm. While this scheme is primary designed for complex 3D terrains, it is can be quite costly and impractical for container sensor networks. Wang et al [25] propose a scheme for locating an RFID affixed object in a 3-dimensional space by referencing to a predetermined coordinate system. This work has shown a viable way of solving sensor localization problem in 3D scenarios. However, this system setup – that of deploying reference RFID tags and/or reader array along the floor and ceiling is oriented towards indoor settings and may not be readily applicable in the case of 3D container sensor networks. In addition to the above, Cheng et al [5] studied the 3D localization problem in sparse underwater acoustic sensor networks. Their solution to the problem is based on two assumptions: (i) each sensor node is aware of its depth information (e.g. by using pressure sensors) and (ii) the sparsely populated underwater acoustic sensors could be projected to a corresponding 2D plane and then localized by using acoustic ranging techniques. Both these assumptions are not valid in the case of container sensor networks for the following reasons. A sensor attached to a container need not have information about its height. In addition, the sensor distribution in container sensor networks is quite dense along the three axes implying that it is almost impossible to have a non-degenerative projection along a 2D plane to preserve network localizability. Therefore, the technique proposed in [5] may not be applicable for container sensor networks.

The distributed localization algorithm presented in this paper is devoid of these issues. The proposed scheme leverages on the underlying container network’s structure and the relative differences in the received signal strength between different network nodes.

III. A LOCATION MANAGEMENT SCHEME FOR 3D CONTAINER SENSOR NETWORKS

A. System Set-up

Figure 1 shows the picture of a typical storage yard in a port where thousands of containers are stacked. Each container is assumed to be affixed with a sensor equipped with a ZigBee like radio. The sensors monitor the status of the containers. While the sensor can be located anywhere inside the container, the antenna is assumed to be fixed outside the container (and wired to the sensor) in a way that data transmission and reception is not affected1. The entire container sensor network is assumed to be connected. When the container monitoring network reports a suspicious activity near the vicinity of specific container(s), human personnel can be deployed for further inspection of the container(s).

![Real world container yard](image)

Standard closed containers come in two possible sizes: \((l \times b \times h)\): 19’4” × 7’8” × 7’0”” and 39’5” × 7’8” × 7’0”. In this paper, we assume that the container sensor network consists of containers of the same size. This assumption is valid in most scenarios, since containers of different sizes are likely to be

1The authors have demonstrated the feasibility of such a set-up in real-life to DoT officials in October 2008.
be stored in separate locations in a port yard. From the picture of a container yard shown in Figure 1, we can decipher the following:

- Containers are stacked in approximately a grid like fashion along rows and columns.
- Containers are likely to be stacked in the same orientation.
- Gaps are likely to exist between container rows and columns to allow personnel to physically walk up to a container and inspect it. These gaps also facilitate radio signal propagation between sensors attached to different containers.
- The height to which containers are stacked can vary between different container columns.
- Given a container, the distance to its neighboring containers along the three different axes is likely to be different (due to the physical dimensions and the presence of gaps).

We leverage the above structure in developing a location management scheme for 3D container sensor networks as follows. First, each container is modeled as a cuboid and the container network as a 3D grid of cuboids as shown in Figure 2. Without loss of generality, the X, Y, and Z axes of this 3D grid are respectively assumed to be along the length, width, and height of the containers. Second, each container is assumed to be a point in the 3D grid and occupy one unit along each of the directions. For example, the containers at the bottom along the front row will occupy the locations (from left to right) – (0, 0, 0), (1, 0, 0) and (2, 0, 0). Third, the grid locations related to origin are non-negative. Also, the radio range is assumed to be high enough to reach at least two grid nodes along any axis. This is valid since the range for sensors with ZigBee transceivers (eg. TelosB) is roughly 75m to 100m in outdoor settings and 20m to 30m in indoor settings [23].

Given the above, the objective of the location management scheme proposed in this paper is to determine the location of nodes sequentially in order starting from the grid origin, a.k.a. the anchor node. Then based on the RSSI information and our message transmission protocol, the location of the nodes along the Z direction are first discovered; then the discovery mechanism expands and determines the location of the nodes along the Y axis; once the locations of nodes along the Y axis are determined, the discovery mechanism expands along the X axis. This sequence of location discovery is shown in Figure 3.

C. Algorithm Description

1) Location discovery along Z axis: The general scheme of the sequential location discovery mechanism of our algorithm could be seen as a simplified 3D greedy perimeter geometry routing process in one direction, as in Karp et al [8]. In the following paragraphs, we describe the working of our location management algorithm with reference to Figure 3. The anchor node at location (0, 0, 0) initiates the process by advertising a message. This message contains the identity of the sender, its location (if discovered), and the power with which the message was transmitted. This message will be received by the anchor node’s geographical neighbors (and possibly others in the radio range too), whose locations are unknown.

with an air gap of 5′ between stacks, then the grid location implies that the container is roughly at a distance of \(\sqrt{(120 \times 40')^2 + (45 \times 13')^2}\) from the grid origin and is at a height of 8 × 8′.

To aid us in our location management, we assume that the identity and location of the container at the grid origin is known. This node acts as the anchor node. The location of other nodes are determined relative to the position of this anchor node. We also assume that the sensor node at the bottom of the final stack along the Y axis (i.e., the sensor node at the location (0, N, 0) for maximum possible N), knows that there are no more stacks along the Y axis. This sensor needs not to know its location, but only needs to know that it is the end of Y axis. We refer to this node as the pseudo-anchor node.

B. Overview of the 3D Location Management Scheme

The basic idea behind the proposed location management scheme is to determine the location of nodes sequentially in order starting from the grid origin, a.k.a. the anchor node. Then based on the RSSI information and our message transmission protocol, the location of the nodes along the Z direction are first discovered; then the discovery mechanism expands and determines the location of the nodes along the Y axis; once the locations of nodes along the Y axis are determined, the discovery mechanism expands along the X axis. This sequence of location discovery is shown in Figure 3.
The received signal strength of this message at different nodes is governed by the following equation:

\[
\text{RSSI} = (K \cdot P_0 \times d^{-\alpha}) - \eta \tag{1}
\]

where, \(K\) is a proportionality constant, \(P_0\) is the transmitted power, \(d\) is the geographical distance between source and receiver, \(\alpha\) is the path loss exponent, and \(\eta\) is the fading component. Though the fading component and path loss exponent can exhibit spatial variations, in a given locality (say within the area of few containers), they may not vary significantly. Therefore, from the above equation, it can be claimed that the node geographically closest to the anchor node is most likely to receive the message transmitted by the anchor with the strongest RSSI. Note that the actual value of the RSSI is not being used here; only the relative ordering of the received RSSIs is being used.

On account of the container stacking characteristics, the node directly stacked on top of the anchor (the neighbor along the Z axis) is likely to be the geographically closest node. Consequently, if the anchor node’s neighbors can determine who received the strongest RSSI, then on virtue of the common knowledge associated with the container stacking, that node can allocate itself a grid location of \( (0, 0, 1) \). This node now becomes the new anchor node and the process can be repeated until the location of all the containers stacked on top of the node at \( (0, 0, 0) \) are determined.

In order to implement the above said location discovery along the Z axis, we need a distributed methodology to detect the node with the strongest RSSI. We achieve this through the use of timers. As soon as a neighboring sensor node receives a message from the anchor node, it starts a timer whose value \( T \) is defined by equation 2. In other words, the timer value at a node with a strong RSSI will be less when compared to the timer at a node with a relatively weak RSSI. Consequently, the timer at the node with the strongest RSSI will expire first. As soon as the timer expires at a node, that node declares itself to be the strongest RSSI node and broadcasts another message with its RSSI. Other nodes on receiving this message, cancel their timers.

\[
T = t \times (P_0 - \text{RSSI}) \tag{2}
\]

where, \(t\): a scaling factor.

\(P_0\): transmitted power.

\(\text{RSSI}\): receiver’s RSSI.

The above algorithm will work in the following way in the network shown in Figure 4. The anchor node A, whose location is \( (0, 0, 0) \), sends out a message. Node B will be the one with the strongest RSSI and B is found to be anchor node A’s closest neighbor. The algorithm assigns a 3D coordinate \( (0, 0, 1) \) to B based on A’s location. Now B becomes the new anchor node and repeats the same process\(^2\). Node C’s location can be found as \((0, 0, 2)\), and D: \((0, 0, 3)\), and F: \((0, 0, 4)\). However, it’s not hard to notice that F’s actual coordinate should be \((0, 1, 1)\). It is clear that the procedure described so far works well only for nodes along the Z axis. We need additional mechanisms to detect the strongest RSSI from nodes in neighboring stacks – i.e., we need ways to detect changes in Y and X co-ordinates.

2) Location Discovery along Y axes: In order to detect changes in Y and X co-ordinates, we introduce two node labels. First, we label a node that receives the strongest RSSI from anchor node as a temporary anchor node temp–anchor \(^3\). In other words, temp-anchor refers to the fact that a node is the immediate neighbor of an anchor node along the Z axis. Second, as mentioned in the end of section III-A, we label the node in the bottom of the farthest stack along the Y axis as the pseudo anchor node pseudo – anchor. This label is manually assigned to a single node alone and is a permanent one (the grey node in Figure 2). Note that the locations of both temp-anchor and pseudo-anchor nodes are not known.

The following steps describes our proposed algorithm with reference to Figure 4.

1) The anchor node A with location \((0, 0, 0)\) sends out a message. Using the timer mechanism discussed before, node B finds itself to be the one with the strongest RSSI. Instead of classifying B to be a discovered node with location \((0, 0, 1)\), B call’s itself a temp – anchor node. It then sends out another message similar to the one sent by A. Node C will posses the strongest RSSI for B’s message and will respond back to B with its own RSSI. Since B’s distance to A is roughly equal to C’s distance to B, based on equation 1, B’s RSSI from A should be more or less the same as C’s RSSI within a pre-set tolerance limit. This fact is verified to make sure that node B is not the top node in the container stack. After verification, the system assigns a coordinate \((0, 0, 1)\) to B and sets it to be the new anchor node.

2) This process is repeated and C can discover its coordinate to be \((0, 0, 2)\).

3) When C repeats the process, D becomes the temp – anchor node. When D sends out a message, F will be the node with the strongest RSSI. However, F’s distance

\(^2\)Nodes with known locations do not participate in the process and hence will not compete for the strongest RSSI title.

\(^3\)This label is assigned by the algorithm and exists only when the algorithm is running. This label also keeps moving from node to node.
to its sending node D is greater than D’s distance to its sending node C. Consequently, there should be considerable difference between D’s RSSI and F’s RSSI. Based on this difference, D understands that it is the top node in the stack.

4) Then the algorithm assign D a coordinate (0, 0, 3). After this D sends a message to the grid origin A (possibly through other nodes below it) stating that the entire stack has been discovered. Now, A becomes the anchor node and sends another message. In the message it indicates that the strongest RSSI node will be along the Y axis. A’s neighbors along the Z direction (B, C, and D) all have their locations discovered and hence do not respond to this message. Through the timer mechanism, node E finds itself to have the the strongest RSSI. The system assigns E a coordinate (0, 1, 0) and sets it to be the anchor node of the next stack in Y direction. Thus the system automatically detects the next stack along the Y axis. Then E can repeat the whole process to identify F’s location as (0, 1, 1).

Algorithm 1 describes these steps in the form of a pseudo code.

3) Location Discovery along X axis: The algorithm is guided to discover stacks along the X axis by the pseudo – anchor node. To illustrate how the pseudoanchor node works to guide the algorithm goes to the next stack in the X direction, let us refer to the Figure 5, where nodes A, B, C form the front stack and nodes D, E form the back stack in Y direction, G, F form the next stack in X direction, and H, I, J form another stack.

Algorithm 1 Locate the Next Sensor Nodes in Z and Y axis

Require: Initialization
1: Initialize every node in the container sensor networks
2: if AnchorNodes or PseudoAnchorNode then
3: IsItBottom = true
4: IsItFound = true
5: end if
6: if AnchorNode then
7: IsItFrontNode = true
8: SendingNode = AnchorNodeID
9: BottomNode = AnchorNodeID
10: end if
11: if PseudoAnchorNode then
12: IsItBackNodes = true
13: end if
14: \((x, y, z) = \text{SendingNode} \) location.
15: if \(\text{SendingNode} \) then
16: Send a message with transmit power \(P_0\), and \((x, y, z)\) information.
17: end if
18: Check whether received a message AND their location is unknown or not
19: if Nodes received message AND their location is unknown then
20: Calculate logic delayed time based on Equ 1, 2:
21: \(\text{DelayedTime} = t \times (P_0 - \text{RSSI})\)
22: \(\text{CurrentTime} = T_0\)
23: Wait until \(\text{CurrentTime} = T_0 + \text{DelayedTime}\)
24: Set the node whose timer expires first as Temp Anchor Node
25: Broadcast a message with its RSSI information: \(\text{tempRSSI} = \text{RSSI}\)
26: Go through lines 14 to 23
27: Identify the closest node and read its RSSI: \(\text{closeRSSI} = \text{RSSI}\)
28: end if
29: if \(\text{closeRSSI}\) is close to \(\text{tempRSSI}\) then
30: Replace Temp Anchor Node flag with new Anchor Node and location found
31: Set its 3D coordinates: \((x, y, z + 1)\)
32: else
33: Set Anchor Node the bottom anchor node of the Z axis and restart the process
34: Go through lines 14 to 23
35: Replace Temp Anchor Node flag with new Anchor Node and location found
36: Set its 3D coordinates: \((x, y + 1, z)\)
37: Set new anchor node bottom node
38: end if

1) Based on the mechanism described in Algorithm 1, we can discover the location of the following nodes B: \((0, 0, 1)\), C: \((0, 0, 2)\), D: \((0, 1, 0)\).

2) Now E is the temp anchor node, after node E sends a message and identifies a lower strongest RSSI value from node I, it knows that it’s a top node of the back stack (pseudo anchor node D identifies that it’s a back stack), the system will then assign E: \((0, 1, 1)\). Then the pseudo-anchor node D will discover its second closest neighbor (apart from E) H and label it as the pseudo-anchor node on the next stack. Node D then sends a message all the way back to the origin A saying that discovery along the Y axis is completed.

3) Now, A becomes the anchor node and sends another message. In the message it indicates that the strongest
RSSI node will be along the X axis. A’s neighbors along the Z direction (B, C, and D) and Y direction (D) have their locations discovered and hence do not respond to this message. Through the timer mechanism, node F finds itself to have the the strongest RSSI. The system assigns F a coordinate (1, 0, 0) and sets it to be the anchor node of the next stack in X direction and the entire process repeats itself.

IV. EXPERIMENTAL RESULTS
A. A 3D Container Sensor Networks Simulator
In view of the limitation of such popular simulators as TOSSIM and ns-2 in simulating a 3D sensor network, we developed our own simulator using NetLogo [27]. Our simulator simulates the 3D container sensor network environment as well as the proposed 3D location discovery scheme. In our experiments, each container sensor node is modeled as a sphere, and the distance between each node in Z axis: 2m, in Y axis: 4m, in X axis: 6m. Note that we don’t need a very accurate waveform, transmission model, and decay factor to simulate our location discovery scheme as long as the RSSI is proportional to distance a.k.a. no absolute RSSI value is needed.

![3D CSN Simulator: after our 3 location management scheme](image)

After initializing the 3D container sensor networks simulator, we will be able to see a 3D red and green sphere array. Red spheres represent the location-unknown container sensor nodes, while green spheres represent the front and back anchor nodes. After the simulation of our 3D location management scheme is done, it looks like Figure 6, (though there will be white spheres represent those sensor nodes that can receive message from anchor nodes, yellow spheres represent those temp anchor nodes during the discovery process) and links between nodes illustrate the message transmission scheme.

B. Results and Analysis
We have carried out extensive experiments in our 3D container sensor networks simulator to study the behavior of our 3D location management scheme and message transmission mechanism. In particular, we studied the performance of our algorithm with respect to the following parameters:

- Total messages sent.
- Total time for algorithm to complete.
- Robustness to spatial variations in fading/shadowing.

1) Message complexity and Completion time: In our experiments, we set the threshold resolution to be 40 percent of the minimum resolution required to distinguish the next non-Z axis neighbor. We ran our location management scheme in 3D container sensor networks of different sizes – from $2 \times 2 \times 2$ to $10 \times 10 \times 10$ (i.e., from 8 nodes to 1000 nodes). We observed the total number of messages exchanged by all the nodes and the time taken for the algorithm to complete. In those experimental results, we can see that spatial variations in shadowing were assumed to be negligible, as evidenced by the identical curves even when up to 15% noise (which is less likely to happen in real world) is presented. The results our experiments are shown in Figure 7(a) and Figure 8(a).

From these figures we can observe that both the message complexity and the completion time of our algorithm is linear with respect to the network size. Consequently, we can deduce that the proposed location management scheme is highly scalable. This is verified by the Figure 7(b) and Figure 8(b). These figures show that the number of messages sent per node and the average time taken to detect the location of a given node reach a saturation value as the network size is increased.

2) Robustness to spatial variations in shadowing: The proposed location management scheme works by leveraging on the perceived variations in RSSI between nodes at different locations. Unlike other RSSI based schemes, the proposed scheme is not sensitive to the absolute value of the RSSI but to the relative differences in RSSI. Therefore, the proposed scheme will be robust to the variations in RSSI caused by shadowing/fading but will be sensitive to the spatial variations in the shadowing/fading effects. While shadowing/fading effects can vary spatially, such effects on the algorithm will be limited. This is because, the proposed location management scheme will be sensitive to such variations only if they occur in the grid neighborhood of a container which may not have a radius of more than 10 metres.

Nevertheless, we studied the sensitivity of the proposed location management scheme to spatial variations in shadowing/fading. We performed this experiment by artificially altering the parameter $\eta$ in equation 1. The parameter $\eta$ was varied to be from 0% to 20% of the received RSSI value. For different values of $\eta$, we observed the error rate of the algorithm – i.e., we observed the fraction of nodes whose location was incorrectly identified. The results of our study are shown in Figure 9(b).

From the figure, we can notice that for small shadowing variations, the algorithm works quite well, i.e., there are hardly any nodes with incorrect locations. However, as the shadowing variations increase to the unusual level, such as 20%, we see that the algorithm begins to assign incorrect grid locations to the sensor nodes. The number of incorrect assignments is not large – at most 8.5%. Figure 9(a) shows another side-effect of large shadowing variations. The figure shows the number of sensors that were not assigned any location at all by the algorithm. In other words, Figure 9(a) shows the number of nodes that were missed assigning a
Fig. 7. Number of messages

(a) Total

(b) Average

Fig. 8. Delayed Time

(a) Total

(b) Average

Fig. 9. Noise effect on a $6 \times 6 \times 6$ container sensor network.

(a) Missed Nodes

(b) Mis-identified Nodes
location by our algorithm. The reason these misses happen is because when an anchor node sends a message, due to the spacial shadowing/fading variations, a geographically farther node might declare that it has stronger RSSI than closer node. In a distributed setting, if such case happens many times, it’s likely that some nodes will never be able to have the opportunity to declare it has the strongest RSSI during the whole message transmission cycle, even if they were the true neighbors of the sending nodes. However, we can see that the proposed algorithm is quite robust in the sense even for shadowing/fading variations as high as 20%, only less than 2% of the nodes are missed.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a distributed location management scheme for 3D container sensor networks. The proposed scheme leverages on the underlying container network’s structure and the relative differences in the received signal strength between different network nodes. A distinguishing feature of the proposed work is that unlike most of the RSSI based algorithms, our algorithm is not required to use the absolute value of RSSI and thus is more robust to shadowing and path loss variations. Performance studies undertaken using a rudimentary 3D container sensor network simulator show that the proposed localization scheme works satisfactorily and has acceptable time and message complexities.

Meanwhile, as sensor networks application continues to evolve, it’s not unusual to envision the 3D application scenario, such as our container sensor networks application. We strongly feel that there is a need for developing a comprehensive 3D sensor network simulator to further help extending the current 2D localization paradigm into potential 3D applications.

Acknowledgment: This work was partly funded by grants from the following organizations: US Federal Highway Administration under grants SAFETEA-LU 1934 and SAFETEA-LU 1702, National Science Foundation under grant CNS-0709329, and Oklahoma Center for the Advancement of Science and Technology.

REFERENCES

Distributed Topology Control for Efficient OSPF Routing in Multi-hop Wireless Networks

Guangyu Pei†, Xiaogang Yang‡, Xufei Mao‡, Wen-Zhan Song‡

Department of Computer Science, Illinois Institute of Technology.

Email: Guangyu.pei@boeing.com, xiaogang.yang@wsu.edu, xmao3@iit.edu, songwz@wsu.edu

Abstract—This paper studies distributed topology control algorithms to support efficient Open Shortest Path First (OSPF) link state routing in multi-hop wireless networks. It is highly desirable to retain the basic OSPF model of reliable flooding, especially when large quantities of external, rarely-changing routing data must be carried across the radio network. However, it is not easy to implement reliable flooding, and improper implementations may cause excessive message control overheads. An existing patent method [1] proposes an expanding ring algorithm to build adjacency graph for reliable flooding. We first analyze this method and show the message complexity of reliable flooding based on it could be $O(n)$ times of optimum. We then show that a slight modification on it can generate an adjacency graph, called ER-CDS, which supports reliable flooding with message complexity $O(1)$ of optimum. However, the construction cost based on the patent method is still high and does not adapt to network topology changes well. Finally, we propose a localized splash merging algorithm to construct ER-CDS, and conducted extensive simulations to evaluate its performance.

keyword: Topology control, Connected Dominating Set, Reliable Flooding

I. INTRODUCTION

Existing wired communication networks use various algorithms for disseminating routing data necessary for routing packets from a source node to a destination node. Each node of the network that handles packets needs sufficient knowledge of the network topology such that it can choose the right output interface through which to forward received packets. Link state routing algorithms, such as Open Shortest Path First (OSPF) algorithm, permit the construction of a network topology such that any given node in the network may make packet-forwarding decisions.

However, a number of difficulties may arise if OSPF is implemented over a multi-hop, multi-access packet radio network with its own private, internal routing system. Although the private, internal routing system of such a network may enable it to appear to be much like a wired network, the properties and characteristics of such radio networks are nevertheless much different from those of wired point-to-point or multi-access networks. Likewise, the adoption of the standard OSPF point-to-multipoint network model for distribution of routing information over a multi-hop routing network is equally inappropriate. The links employed by a point-to-multipoint model to represent the radio networks are likely to be several radio hops in length, and so their use to distribute routing information would often require packet replication and/or result in transmitting duplicate information over a single radio link. Furthermore, the network of links needed by a point-to-multipoint model to represent the radio network may be much more dense than required for distribution of routing information. Finally, this network of links may be constantly changing in response to the need for accurate representation of the radio network, and so may not be sufficiently stable for effective use in distributing routing information.

On the other hand, it is highly desirable to retain the basic OSPF model of reliable flooding, especially when large quantities of external, rarely-changing routing data must be carried across the radio network. Therefore, there exists a need for systems and methods that can solve some of the inherent problems that exist with distributing OSPF routing information across a multi-hop, multi-access packet radio network, while maintaining full compatibility with standard OSPF over the networks and preserving the basic OSPF model of reliable flooding.

An existing patent [1] proposes an expanding ring algorithm to build adjacency graph for reliable flooding. We first analyze this method and show the message complexity of reliable flooding based on it could be $O(n)$ times of optimum where $n$ is the network size. We then show that a slight modification on it can generate a adjacency graph, called ER-CDS, which supports reliable flooding with message complexity $O(1)$ of optimum. However, the construction cost based on the patent method is still high and does not adapt to network topology changes well. Finally, we propose a localized splash merging algorithm to construct ER-CDS, and extensive simulation results show that our ER-CDS has better performance than others in most aspects. ER-CDS stands for Expanding Ring based Connected Dominating Set. In other words, our proposed splash merging algorithm will form a Connected Dominating Set (CDS) to support reliable flooding of OSPF routing information. Basically all nodes will be divided into three groups, dominators, connectors and dominatees, and all dominators and connectors will form the CDS and can be considered as the virtual backbone (infrastructure). To disseminate the OSPF routing information, the link state packets originated from a dominatee will first be forwarded to its
dominator, which then broadcasts to all its dominatess and adjacent dominators in CDS (through connectors), until all of them confirm the reception. Every adjacent dominators will repeat this process until reliable flooding to whole network is accomplished. In other words, only dominators and connectors will participate message re-broadcasting, which significantly reduces the message overheads of flooding. In the literature, the application of CDSs in the wireless network have been widely studied. For example, [4], [5] formed an underlying architecture used by protocols including media access coordination; Datta and Sto in [6] used CDS to do location-based routing; the issues related to energy conservation have been studied in [7]–[9]; and Budhaditya et al., and Ding et al., studied how to use CDS to do topology control in [10] and [11] respectively. Actually many works seek a Minimum Connected Dominating Set (MCDS) in unit-disk graphs as their major design goals. This is because the system control-related messages from the base station or sink node to all wireless nodes will count on the virtual backbone (nodes on the CDS). Thus performance bounds is directly related to the properties of the CDS. Intuitively, minimizing the cardinality of the computed CDS can help to decrease the control overhead for most of application, like broadcasting for route discovery in [12], [13]. However, constructing MCDS is a well-known NP-hard problem. In this paper, we will show that the ER-CDS constructed by our splash merging algorithm has the following properties: (1) The CDS size is bounded by 2|MCDS|. (2) The time and message complexity of splash merging algorithm is \(O(|DS|)\) and \(O(n + |DS||CDS|)\) where \(|DS|\) and \(|CDS|\) is the cardinality of DS and CDS of ER-CDS.

The rest of the paper is organized as follows. In section II, we analyze an existing patent method and propose an improved structure, called ER-CDS, to guarantee worst-case performance. In III, we propose an innovative splash merging algorithm to construct ER-CDS, with low message complexity and high adaptivity to topology changes. We conducted extensive simulations on TOSSIM 2.0 and compared our algorithms to other CDS algorithms. Simulation results are shown in section IV. Related works have been presented in section V. Finally, we conclude the work in section VI.

II. Expanding Ring Graph Analysis

A multi-hop wireless ad hoc network is modelled by a set \(V\) of \(n\) wireless nodes distributed in a two-dimensional plane. Each node has the same maximum transmission range. By a proper scaling, we assume that all nodes have the maximum transmission range equal to one unit. These wireless nodes define a unit disk graph \(UDG(V)\) in which there is an edge between two nodes iff the Euclidean distance between them is at most one unit. In other words, we assume that two nodes can always receive the signal from each other directly if the Euclidean distance between them is no more than one unit. Notice that, in practice, the transmission region of a node is not necessarily a perfect disk. As done by most results in the literature, for simplicity, we model it by disk in order to first explore the underlying nature of ad hoc networks. It has been proved in [30] that for a wireless networks with \(n\) nodes falling into a unit square, when \(\pi nr^2 \geq \log n\), the network is connected with high probability where \(r\) is the transmission range of the ordinal wireless nodes. Clearly, by a proper scaling of the transmission range (one unit as aforementioned) and the the side length of the square region, we can have a connected wireless network with high probability. Hereafter, for simplicity, \(UDG(V)\) is always assumed to be connected. We also assume that all wireless nodes have distinctive identities (IDs).

In the patent [1], an expanding ring algorithm is proposed to construct adjacency graph for reliable flooding of OSPF link state packets. The algorithm works as following: each node \(u\) tries to find another node with smaller ID in its \(k\)-hop neighborhood, with \(k\) starts from 1 and increments in each round, until node \(u\) finds another node \(v\) such that \(ID(v) < ID(u)\) or it has reached all nodes in the network. The later case happens when node \(u\) has the smallest ID. If node \(u\) finds another node \(v\) such that \(ID(v) < ID(u)\), then we denote this relationship as \(u \rightarrow v\) and called \(u\) has an out-edge while \(v\) has an in-edge. It is not hard to see that the resulting graph, called Expanding Ring (ER) graph, is a shortest path tree rooted at the node with smallest ID. This exemplary technique creates a spanning forest, with each tree in the forest comprising a shortest-path tree. Each of the shortest-path trees can then be linked together to form a tree rooted at the lowest numbered router. The OSPF broadcast performs by asking every non-leaf node to rebroadcast the message to its neighbors in the tree until all neighbors confirm it.

**Lemma 1:** The message complexity of broadcast based on Expanding Ring (ER) graph can be \(O(n)\) times of the optimum, here \(n\) is the number of nodes in the network.

**Proof:** Figure 1 illustrates such a possible network topology: assume each node’s transmission range is \(r\), the inner circle’s radius is \(r\), and outer circle’s radius is \(2r\); there are \(2n+1\) nodes in the network, the center node ID is 0 (assume the node who has smallest id is 0), while the node IDs in inner and outer circle is \([1, n]\) and \([n+1, 2n]\) respectively.

According to the expanding ring algorithm, every node in outer circle with ID \(n+i\) has an out-edge to a node in the inner circle with ID \(i\), and every nodes in inner circle has an out-edge to the center node. In other words, the Expanding Ring graph consists of \(n\) rays centered at node 0, as illustrated in Figure 1 (b). Hence, The broadcast operation can be implemented by asking the node 0 and all nodes in inner circle to re-broadcast, while the outer circle nodes are not. Consequently, for optimal, center node only needs to broadcast once, then inner circle nodes need to broadcast once for each, so the total broadcasting cost is \(n + 1\).

On the other hand, the Minimum Connected Dominating Set (MCDS) on the same topology includes node 0 and \(\pi/\arcsin(1/4)\) number of nodes at the outer circle as dominators, and \(\pi/\arcsin(1/4)\) number of nodes at the inner circle as connectors, as illustrated in Figure 1 (a). The broadcast operation can be implemented by asking every node in CDS
The size of ER-CDS is bounded by $12 \cdot |OPT| + 1$, where $OPT$ is the size of MCDS (Minimum Connected Dominating Set).

Proof: If there is a dominator of ER-CDS in OPT, then following similar proof of Lemma 3, we can show that the total number of dominators is at most $1 + 4(|OPT| - 1) = 4|OPT| - 3$. Because ER-CDS is a tree and there are at most two connectors between adjacent dominators, the number of connectors is at most $2 \cdot 4(|OPT| - 1) = 8|OPT| - 8$. Hence $|ER-CDS| \leq 12|OPT| - 11$. 

---

Fig. 1. Broadcast cost of Expanding Ring graph can be as expensive as $O(n)$ times minimum broadcast cost.
Now, we assume there is no dominators in OPT. Let $k$ be the number of dominators adjacent to the root of OPT. Similarly as the proof of lemma 3, we can show that $k \leq 5$ and the total number of dominators in ER-CDS is at most $k + 4(opt - 1)$. Again, because ER-CDS is a tree and there are at most two connectors between adjacent dominators, the number of connectors is at most $2(4opt - 5 + k) = 8opt - 10 + 2k$. Hence $|ER - CDS| \leq 12opt - 14 + 3k \leq 12opt + 1$.

Overall, $|ER - CDS| \leq 12opt + 1$.

Notice that, smaller CDS means lower broadcast message complexity since nodes in CDS needs to rebroadcast messages.

III. SPLASH MERGING ALGORITHM FOR ER-CDS CONSTRUCTION

With the slight modification on the original expanding ring algorithm, the resulting ER-CDS graph has constant bounded backbone size. However, the ER-CDS construction based on modified expanding ring algorithm (as described in previous section) is still expensive. Some dominator node may need to broadcast multiple rounds of search messages before it finds another dominator with smaller ID. An example of expanding ring procedure is illustrated in Figure 2. It takes 10-hop expanding-ring search before the ER-CDS forms, and 4,6,7,8,9 hops search did not change anything except wasting energies.
In the section, we introduce an innovative splash merging algorithm to reduce the communication cost of constructing ER-CDS. In addition, it is not difficult to see that it also works for ER-CDS updates when topology changes, as it is a localized algorithm. The basic idea is to keep merging splashes (e.g., clusters) to bigger splashes, until it covers the whole network.

The following notations will be used in the rest:

1) \( \text{Size}(u) \) is the number of dominatees of a dominator node \( u \).
2) \( \text{Head}(u) \) is the splash (e.g., cluster) head of node \( u \).
3) \( \text{OldHead}(u) \) is the previous splash (e.g., cluster) head of node \( u \).
4) \( \text{AdjHead}(u) \) is the adjacent splash head of node \( u \). It applies when node \( u \) is a potential connector.
5) \( \text{AdjNode}(u) \) is the adjacent nodes of node \( u \) which belong to an adjacent splash. Both \( u \) and \( \text{AdjNode}(u) \) could be potential connectors.
6) \( I - \text{AM} - \text{DOMINATOR} \) is the message sent by the node which becomes dominator.
7) \( I - \text{AM} - \text{DOMINATEE} \) is the message sent by the node which becomes dominatee.
8) \( I - \text{TRY} - \text{CONNECTOR} \) is the message sent by the node that is the potential connector.
9) \( \text{UPDATE} - \text{HEAD} \) is used to merge splash into new splash.

Algorithm 1 gives the splash merging algorithm details. Figure 4 illustrates the procedure of splash merging algorithm on same network topology in Figure 2.

**Theorem 5:** The message and time complexity of ER-CDS formation algorithm 1 is \( O(n + |DS||CDS|) \) and \( O(|DS|) \) respectively. Here \( |DS| \) and \( |CDS| \) is the size of Dominator Set and Connected Dominating Set respectively.

**Proof:** In Step 1 of algorithm 1, each node sends two messages: one announces its ID, another announces its role as \text{DOMINATOR} or \text{DOMINATEE}. That is to say, the message and time complexity are \( O(n) \) and \( O(1) \) respectively.

In Step 2 of algorithm 1, the communication is along the CDS tree of each splash. In other words, only CDS nodes need to forward packets in each splash. And those CDS nodes participate in message forwarding for at most \( |DS| \) times, as it takes at most \( |DS| \) steps to connect all dominators. Hence, the
Algorithm 1 Splash Merging Algorithm for ER-CDS Construction

Step 1: Find dominators based on smallest ID.

1) Initially, every node \( u \) is marked as UNDECIDED, and set \( \text{Size}(u) = 0 \), \( \text{Head}(u) = \text{OldHead}(u) = \text{AdjHead}(u) = \text{AdjNode}(u) = u \) and broadcast its own ID \( u \).

2) If an UNDECIDED node \( u \) has smallest ID among its UNDECIDED neighbors, it will mark itself as DOMINATOR and broadcast a message I-AM-DOMINATOR(\( u \), Head(\( u \))).

3) If an UNDECIDED node \( v \) receives a message I-AM-DOMINATOR(\( u \), w) and Head(\( v \)) \( > u \), then node \( v \) will mark itself as DOMINATEE and record its dominator Head(\( v \)) = \( u \), then broadcasts a message I-AM-DOMINATEE (\( v \), Head(\( v \))).

4) If a DOMINATOR node \( u \) receives a message I-AM-DOMINATEE (\( v \), \( x \)) and \( x \) \( = u \) for the first time, then it sets its \( \text{Size}(u) = \text{Size}(u) + 1 \).

Step 2: DOMINATOR node runs expended ring algorithm to search connectors, until it finds and updates to C-DOMINATOR (e.g., Connected Dominator).

1) Event: a node \( v \) receives a message I-AM-DOMINATOR(\( u \), w) or I-AM-DOMINATEE(\( u \), w) or UPDATE(\( u \), w, o, y, z) (if \( o \neq \text{headID}(v) \)) from \( u \).
   - If \( w < \text{AdjHead}(v) \) and \( w < \text{Head}(v) \), then it sets AdjHead(\( v \)) = \( w \), and AdjNode(\( v \)) = \( u \). Basically, a node remembers potential connectors to the closest splash (e.g., cluster) with smallest head ID. If a node \( v \) finds its AdjHead(\( v \)) < Head(\( v \)), then sends a message I-TRY-CONNECTOR (\( v \), Head(\( v \)), AdjHead(\( v \)), AdjNode(\( v \))) towards the splash head.
   - If Head(\( v \)) \( = u \) and \( w < \text{AdjHead}(u) \), then it sets AdjHead(\( v \)) = \( w \);
     - if \( v \neq \text{Head}(v) \), send a message I-TRY-CONNECTOR (\( v \), Head(\( v \)), AdjHead(\( v \)), y, z);
     - if \( v = \text{Head}(v) \), set OldHead(\( w \)) = Head(\( w \)), Head(\( v \)) = \( w \), updates to C-DOMINATOR and broadcasts a message UPDATE-HEAD(\( v \), Head(\( v \)), OldHead(\( v \)), y, z).

2) Event: a cluster head \( v \) receives a message I-TRY-CONNECTOR (\( x \), u, w, y, z).
   - If Head(\( v \)) \( = u \) and \( w < \text{AdjHead}(u) \), then it sets AdjHead(\( v \)) = \( w \);
     - if \( v \neq \text{Head}(v) \), send a message I-TRY-CONNECTOR (\( v \), Head(\( v \)), AdjHead(\( v \)), y, z);
     - if \( v = \text{Head}(v) \), set OldHead(\( w \)) = Head(\( w \)), Head(\( v \)) = \( w \), updates to C-DOMINATOR and broadcasts a message UPDATE-HEAD(\( v \), Head(\( v \)), OldHead(\( v \)), y, z).

3) Event: node \( w \) receives a message UPDATE-HEAD(\( x \), u, o, y, z).
   - If Head(\( w \)) \( = o \), then it sets OldHead(\( w \)) = Head(\( w \)), Head(\( w \)) = \( u \);
     - if Head(\( w \)) \( = o \), \( w = y \) and \( w \) is DOMINATEE, then it sets itself as CONNECTOR;
     - if Head(\( w \)) \( = o \), \( w = z \), \( w \) is DOMINATEE, then it sets itself as CONNECTOR. Broadcasts a message UPDATE-HEAD(\( w \), Head(\( w \)), OldHead(\( w \)), y, z).

Finally, the C-DOMINATOR and CONNECTOR forms the backbone of a network, called ER-CDS in the following presentation.

We conduct extensive simulations to evaluate the performance of ER-CDS via TOSSIM on TinyOS 2.0, by comparing it to the algorithm in [27].

A. Simulation Environment

We use TinyOS-2.x TOSSIM to conduct the simulations. To compare the various performances between ER-CDS and Wan’s algorithm in [27], we varies the network size from 30 to 210 with step 30, while keeping the network density unchanged. For each pair of nodes which fall in the transmission range of each other, we randomly generate the link quality between them in order to simulate the realistic situation. The evaluation compared the following four important performance metrics:

- Dominator Size: the number of dominators
- Connector Size: the number of connectors
- Setup Cost: the number of messages used to construct the CDS.
- Broadcast Cost: the number of messages for reliable flooding of link state packets.
As mentioned at the beginning of this paper, OSPF has conventionally been implemented in wired point-to-point or multi-access networks. And to implement OSPF over a multi-hop, multi-access packet radio network with its own private, internal routing system, so as to permit seamless integration of such a network into an OSPF environment. However, lots of problems raise when it is moved to wireless network. Our target is to build a CDS, and realize reliable flood in this network through backbone composed of dominators and connectors. Then we collect and count the cost as following. Dominators and connectors rebroadcast packets to ensure that packets can reach every node in the network, while dominatees only receive packets but not forward them. After CDS is built, the dominators will remember the information of their neighbors, which includes some useful properties, such as how many dominatees and connectors connected to them. Hence, they can make broadcast reliable by checking whether ACKs are received from intended neighbors - if not, a dominator shall rebroadcast again periodically.

**B. Simulation Results**

We evaluate two protocols under same topologies with same density. Figure 5(a) shows that when the network size is not too big, the dominator size of Wan’s Algorithm goes smaller than that of ER-CDS. With the increment of the network size, dominator size increases for both of algorithms, but ER-CDS increases slower than CDS. So at this point, ER-CDS performs better. The reasons for this difference are observed as follows. The way to choose Dominators has been described in the step 1 described in the algorithm 1. For ER-CDS, a dominator is decided by its ID, while a dominator is decided by nodes’ levels (which are determined in the neighbor discovering step) in Wan’s algorithm [27]. After the neighbor discovery, the root node begins to use coloring method to find dominator. For ER-CDS, the uncolored node with smallest ID is more possible to be selected as a dominator, and for Wan’s algorithm, the node with lowest rank \((level, ID)\) has bigger chance to become a dominator. As we can see from the Fig. 5(a), ER-CDS has more advantages When it comes to bigger network size.

Figure 5(b) shows that the trend of connector size with the increment of the network size. As illustrated in the Fig. 5(b), the ER-CDS’s connector size is bigger than the other one. This meets our conjecture and we analyze the reasons for this as follows. The connector size for a graph is directly related with the dominator size. After the dominators are chosen, the dominator will choose some dominatees to form a connected dominating set. According to Wan’s algorithm, they have smaller connector size since the construction of CDS is
based on a tree structure and they guaranteed that there is at most one connector for adjacent dominator pair.

Figure 5(c) shows that setup cost (total messages used to construct CDS) of both algorithms. The setup cost used by ER-CDS is bigger than that of Wan’ algorithm. This is because ER-CDS chooses dominators and connectors by two steps while the dominators and connectors are chosen in the same step in Wan’s algorithm. In the first step, for both algorithm, all the nodes discovery their neighbors first by collecting information from one-hop neighbors. According to Wan’s algorithm, it will find dominators and connectors at the same time since every non-leaf dominatee node will be selected as a connector automatically based on the tree structure; however, for ER-CDS, it only can find dominators by smallest IDs during the first step. In the second step of ER-CDS, every node will update its HeadID round by round until all nodes form a connected single cluster. This part of cost leads to bigger setup cost of ER-CDS. One thing needs to mention that in the implement of Wan’s Algorithm, it assumes the node which has smallest ID will be selected as a root node. This is not true in most of cases and to choose the root also consumes setup resource. This is because that every node have to broadcast information (messages) in order to know the information of the whole network. It has been proved in [31] that for a wireless network with size $n$, to find the root node (or say electing the leader) has message complexity $O(n^2)$ in worst case and $O(n \log n)$ message complexity in average case. The total construction cost of Wan’s algorithm including leader election is also illustrated in Fig. 5(c).

The trend of broadcast cost for both algorithms has been shown in Fig. 5(d). From the result, we can see that with the increment of the network size, ER-CDS has less broadcast cost than the other does. The measurement of broadcast cost is conducted as follows. First the root start to broadcast a packet, then for each dominator or connector, it will rebroadcast the packet after it received the packet. When a dominatee receives a packet, it will send ACK back to the dominator who sends the packet. Next, only if a node finds that it has received all the ACKs from the nodes that connect to root through it, it will reply ACK back to upper layer. Repeatedly until all node has a copy of the packet.

Clearly, the broadcast cost depends on dominator size and original topology. Because ER-CDS selects dominators based on IDs, the topology and ID distribution have big effect on the broadcast cost. One of methods to reduce the cost is to adjust the node ID such that decreasing the dominator size. For example, if we have a simple network with size 3 which forms a line graph, in which node 2 is the intermedia node between node 1 and 3. Based on ER-CDS discipline, we will choose node 1 and 3 as dominators. However, if we could change node’s ID, for example we switch IDs of node 1 and 2, then the only dominator will be node 1. This is an advantage of ER-CDS since Wan’s Algorith [27] didn’t only depends on nodes’ ID. In addition, for Wan’s algorithm, the cost of finding proper root and the dominators is fixed, but in ER-CDS, there maybe some possibility to change partial topology such that the whole network dominator size is decreased.

One thing needs to mention that we did not run simulation to compare our algorithm with the patent method in [1] since its communications cost is clearly higher than our ER-CDS algorithm. In addition, it needs to include the full path information to locate connectors in the message, which may exceed the message size limit of TinyOS (e.g., at most 128 bytes at 802.15.4 radio).

V. RELATED WORKS

After first noted by Ephremedis et al., in [14] that a CDS can create a virtual net- work backbone for packet routing and control. Many researchers have proposed a bunch of CDS related construction methods and application related algorithms in order to improve the performance for wireless networks by using CDS. Basically, algorithms that construct a CDS in ad hoc networks can be divided into two categories: centralized algorithms that depend on network-wide information or coordination and decentralized that depend on local information only. Guha and Khuller proposed two CDS construction algorithms in their seminal work [15] in 1998 in which they proposed two greedy heuristic algorithms with bounded performance guarantees. In the first algorithm, the CDS is grown from one node outward. In the second algorithm, they first constructed a weighted CDS, and then intermediate nodes are selected to create a CDS. In addition, Ruan et al., in [16] gave a one-step greedy approximation algorithm by coloring with performance ratio at most $3 + \ln(\Delta)$. Here, $\Delta$ is the maximum degree of the the communication graph. One of interesting work done by Cheng et al., [17], in which they proposed a greedy algorithm for minimal CDS in unit-disk graphs. Their algorithm relies on an maximal independent set (MIS) but the resultant CDS may not contain all the elements in the MIS. Recently Min et al., in [18] propose to use a Steiner tree with minimum number of Steiner nodes (ST-MSN) to connect a maximal independent set before to compute CDS. However, all of above mentioned algorithm are centralized based which is not suitable for large scale wireless networks, in which decreasing the communication cost is one of important objectives.

In pure localized algorithms (or say distributed algorithms) which is more realistic for wireless networks, the status of each node depends on its $k$-hop topology only, where $k$ is a small constant, and usually converges after at most $k$ rounds of information exchange among neighbors. Chen et al., propose a series of approximate algorithms for computing a small WCDS to be used to cluster mobile ad hoc networks in [19]. Das et al., in [20] implemented both algorithms in distributed pattern proposed by Guha and Khuller in [15]. Alzoubi and Wan’s in [27], [28] utilize the properties of unit-disk graphs (UDGs) in which they provided two versions of an algorithm to construct the dominating set for a wireless network. In both algorithms, they first construct a rooted spanning tree from the original network topology using the distributed leader election algorithm. Then, based on the levels of nodes in the tree, they divided nodes into different ranks and further get a CDS
with the time and messages complexity $O(n)$ and $O(n \log n)$ respectively. In their scheme, a maximal independent set (MIS) is elected such that each vertex in the MIS can be connected to the spanning tree via an extra vertex. Since in unit disk graphs, the size of an independent set is at most 4 times that of the minimum CDS, this algorithm has an approximation ratio of 8. However, this algorithm usually produces a larger CDS than the MCDS algorithm in random unit disk graphs. Later, Cheng introduced two algorithms for growing a connected dominating set from a leader node in [21], [22]. Compared with the work of Alzoubi and Wan’s works in [27], [28], they introduce a new active state for vertices to describe the current labelling set of vertex nodes.

Another type of well known construction methods of CDS is based on pruning. Wu et al., proposes a completely localized algorithm to con- struct CDS in general graphs in [23]. In other works, like in [23]–[26], each node determines its own status and is in the forward status by default. A node resigns its role (dominators, connectors and dominantees) of forward status by itself if a path from the source can be found for each of its neighbors. Nodes in such a path can be either already forwarded nodes or nodes that deem to forward.

VI. CONCLUSION

In this paper, topology control algorithms to support efficient OSPF link state routing has been studied. An existing patent method [1] proposes an expanding ring algorithm to build adjacency graph for reliable flooding. We first analyze this method and shows the message complexity of reliable flooding based on it could be $O(n)$ times of optimum. We then show that a slight modification on it can generate a adjacency graph, called ER-CDS, which supports reliable flooding with message complexity $O(1)$ of optimum. However, the construction cost based on the patent method is still high and does not adapt to network topology changes well. Finally, we propose a localized splash merging algorithm to construct ER-CDS, and conducted extensive simulations to evaluate its performance.

REFERENCES

Two Tier User Authentication Scheme for Heterogeneous Sensor Networks

Le Xuan Hung
Dept. of Computer Engineering
Kyung Hee University
Korea, 446-701
Email: lxhung@oslabs.khu.ac.kr

Sungyoung Lee
Dept. of Computer Engineering
Kyung Hee University
Korea, 446-701
Email: sylee@oslabs.khu.ac.kr

Young-Koo Lee
Dept. of Computer Engineering
Kyung Hee University
Korea, 446-701
Email: yklee@khu.ac.kr

Abstract—For many sensor network applications such as military or homeland security, sensed data is critical that only legitimate users should be allowed to access. Therefore, a user authentication scheme is essential. Existing solutions for computer and ad-hoc wireless networks are not suitable because they are too heavy to deploy on resource-constraint sensor nodes. In this paper, we propose a Two Tier User Authentication scheme (TTUA) for heterogeneous sensor networks. Our analysis and simulation results have shown that TTUA is more secure and energy efficient than existing approaches.

Index Terms—heterogeneous sensor networks, security, user authentication, energy efficient, distributed.

I. INTRODUCTION

Sensor networks have many applications, such as military, homeland security, environment surveillance, manufacturing. Basically, sensed data is collected from the sensor network and presented to users either upon inquiries or upon event detection. For many sensor network applications, the sensed data is critical so that it should be presented only to ‘legitimate’ users. Therefore, an user authentication is very essential.

Due to limitations in power, communication, and computation capacities, applying conventional public key approaches such as RSA is not feasible because they are too heavy for sensor nodes. Symmetric key-based schemes are more suitable. The simplest way is to authenticate user at the base station (BS). However, this is not an efficient way because it requires considerable communication cost and is more time consuming. Recently, several attempts have been made to provide user authentication schemes for sensor networks [5][8][14]. However, they are not energy-efficient and have some drawbacks. For example, the public key approach based on Eclipse Curve Cryptography (ECC) [5] consumes much more energy and computational time than symmetric key-based schemes, approximately 80-fold in energy consumption and 143-fold in computational time; the n-authentication [8] requires many nodes to involve in authentication and communication, leading to high energy consumption and high communication overhead. Those limitations will be discussed in more details in Section II. Most existing work on sensor networks considers homogeneous sensor networks where all sensors have the same capacities in communication, computation, memory storage, power supply, etc. However, it has been shown that homogeneous sensor networks have poor performance and many limitations [9][10]. Recently, Heterogeneous Sensor Networks (HSNs) have become popular, particularly in real deployments because of their potential to increase network lifetime and reliability without significantly increasing the cost [15]. For example, Intel has deployed a pilot application of sensor networks to monitor the health of mechanical equipment in its fabrication plants [15]. In general, a HSN is composed of a large number of homogeneous sensors (e.g. MICA2) along with a small number of additional powerful Personal Digital Assistants (PDAs). Several security studies have explored heterogeneous sensor networks [11][12] to achieve better security and efficiency.

This paper presents our proposed scheme, Two-Tier User Authentication (TTUA), for HSNs. Our contributions are two-fold. First, TTUA is a distributed user authentication scheme. It does not require the involvement of a BS or any center during authenticating process. Second, an careful analysis and simulation are given along with comparison with an existing approach. We analyze and simulate our scheme and compared with existing approaches. Both analysis and simulation results show that TTUA is more secure and significantly reduces energy and time.

The remainder of the paper is organized as following. In Section II we briefly review related work and discuss about their limitations. The system model and assumptions are described in Section III. In Section IV, we present our proposed scheme. The advantages of our scheme are shown through mathematical analysis (Section V) and performance evaluation (Section VI). Finally, Section VII concludes the paper and outlines our future works.

II. RELATED WORK

Recently, several studies on user authentication for sensor networks have been proposed. In general, they can be classified.
into two types: one is based on public key cryptography [5][12][14], and the other is based on symmetric key cryptography [6][7].

In public key-based approaches, Hae-Sung et al. proposed ECC-based access control protocol [5]. Benenson et al. proposed an algorithmic framework for robust authentication and access control in sensor networks [8] which can withstand capture of up to \( t \) nodes. Benenson et al. realized robust user authentication [14] which let the sensors in the communication range of the user serve as interpreters between the "public key crypto world" of the sensor network. Though public key cryptography has been proved feasible on sensors, it is much more resource consuming than symmetric key-based approaches. For example, the ECC-based scheme [5] consumes much energy and computational time than symmetric key-based schemes, approximately 80-fold in energy consumption and 143-fold in computational time.

Few works based on symmetric key cryptography have been proposed. Wong et al. proposed a dynamic user authentication protocol for sensor networks which imposes very light computational load and requires simple operations [6]. However, it is vulnerable from replay and forgery attacks. Also, passwords could be revealed by any of the sensor nodes. Therefore, H.R.Tseng, R.H.Jin, and W.Yang [7] improved Wong et al.'s scheme which not only retains all the advantages in the original scheme but also enhances its security. We name it TJY. However, the scheme relies on a centralized gateway node to perform authenticating process which introduces high communication overhead and prolongs response time. On the other hand, the message ACC_LOGIN between the gateway node and the login-node is not protected. As a consequence, an adversary could easily modify this message to fool the login-node that she is an authenticated one.

III. SYSTEM AND ADVESARIAL MODEL

We consider a large-scale HSN with two types of nodes: a large number of ordinary sensors (e.g. MICA2), and a small number of powerful sensors (called cluster heads - CHs). Ordinary sensors (for short, we call sensors) have a short transmission range, and a small power supply. Due to resource limitation, sensors are not equipped with any tamper-resistant hardware and they are susceptible to node capture attacks. In contrast, CHs have more energy supply, longer transmission range, higher data rate than ordinary sensors. Especially, CHs are equipped with tamper-resistant hardware. This assumption is reasonable because the number of cluster heads in an HSN is relatively small (e.g., 20 cluster heads for 1,000 sensors) so the cost of tamper-resistant hardware is small [11]. We assume that both sensors and CHs are uniformly distributed in the sensor area. After deployment, sensors are grouped in clusters. Each cluster maintains one CH. The CHs form a backbone in the network so that sensed data after being collected will be transmitted through CHs towards the requesting users (see Fig.1). A user may use a powerful computing device, such as a PDA, mobile phone, or laptop, to interact with the sensor network.

The sensor network is managed by a base station (BS), which is responsible for generating all security primitives. For the sake of simplicity, we assume that the BS also acts as a key distribution center. A key pre-distribution operation is carried out by the BS right after the network deployment stage. We can assume that a key pre-distribution scheme [13] is used to distribute symmetric keys to all nodes. The BS maintains a list of pair-wise keys \( K_i \) with each CH, Also, each CH shares a pair-wise key with its member sensor.

An adversary may make use of all possible means to authenticate as a legitimate user. She can capture a small amount of sensor nodes, read out their memory contents to get user authentication information. Through these compromised sensors, the adversary could not only carry out eavesdropping to extract transmitted information, by also could replace some messages to destroy the regular authentication process of a legitimate user. Even more, an authorized user can collude with the compromised sensors to cheat the entire sensor network that he is an authorized one. Also, the adversary can use a powerful device to perform Denial-of-Service (DoS) attacks by sending a large number of authentication requests to a particular node or the whole sensor network.

IV. TWO TIER USER AUTHENTICATION

The Two Tier User Authentication scheme (TTUA) allows a user to register once and authenticate to the network many times. He also can change the password at will. The sequence diagram of TTUA is illustrated in Fig.2. TTUA includes three phases: user registration, user authentication, and password change. In the user registration phase, a user comes to the BS and applies for relevant secret information. In user authentication phase, he uses this information to authenticate a sensor. The user may change his password to protect his authentication account. We discuss each step in TTUA as follows.

A. User registration

Initially, the user comes to the BS to register his ID and password and applies for relevant secret information to his devices via a secure channel. He sends his ID (UID) and password (PW) to the BS. To make password confidential,
it is needed to hash the password using some hash function such as SHA-1 [16] before sending to the BS. Doing so helps to protect the password from being revealed to anyone, even the BS’s administrator. The BS sends a list of pair-wise keys with each CHi to the user’s devices Ks. These pair-wise keys are derived from the pair-wise keys shared between BS and each CHi by using a hash function.

\[ user \rightarrow BS : UID, H(PW) \]

\[ BS \rightarrow user : list\{H(K_i)\} \]

The BS then broadcasts UID and H(PW) to all CHs in an encrypted form.

\[ BS \rightarrow CH_i : E\{UID, H(PW)\}K_i, H(E\{UID\|H(PW)\})K_i \]

where E[M]K is an symmetric encryption using a secret key K, and \| means concatenation.

After verifying the message using its own share pair-wise key K_i, each CHi stores UID and H(PW) in its database.

B. User authentication

Notice that CHs form a backbone so that all communications within the network are relayed through it. Suppose a user wants to access data stored at a sensor s (assume A is the cluster head of s). The authentication process includes the following steps:

Step 1) At first, the user sends his ID (UID1) and his hashed password H(PW1) to A as follows:

\[ user \rightarrow A : UID_1, H(PW_1), T_0, T_0 \]

where T_0 is a current timestamp used to defend against replay attacks, and \( \oplus \) means XOR operation.

At first, the user sends his ID (UID1) and his hashed password H(PW1) to A. A first checks if the timestamp T_0 is valid (T_0 > T_now, where T_now is current time) and UID1 exists in its storage. If yes, then it verifies the password H(PW1) by using H(PW) stored in its database.

A checks:
1. T_0 > T_now, UID_1 = UID
2. H(H(PW_1) \oplus T_0) = H(H(PW) \oplus T_0)

Step2) If the verification is successful, A sends UID, its ID (ID_A), and new timestamp (T_1) along with a MAC using its share pair-wise key (K_{A,s}) with the sensor s.

\[ A \rightarrow s : UID, ID_A, T_1, MAC(K_{A,s}, UID\|ID_A\|T_1) \]

Upon receiving the message, s first checks if T_1 is valid. If yes, it verifies UID and ID_A by generating a MAC with the shared pair-wise key with A (K_{A,s}) and comparing with received MAC. If all of these are successful, then the user is authentic.

After successful authentication, sensor s is ready to send data to the user. s may send a short message to inform user that he is authenticated.

C. Password change

TTIA allows users to change their password at will. He can make it locally via a nearby CH. He can make it done at any location within the network. To do this, he broadcasts his current ID (UID2) and hashed password H(PW2), along
with a new hashed password $H(PW^*)$ to all $CHs$.

$$user \rightarrow \{CH_i\} : broadcast\{UID_2, H(PW_2), H(PW^*)\), MAC(H(K_i), UID_2) || H(PW_2) || H(PW^*)\}$$

Each receiving $CH_i$ computes MAC value and ensures that $UID_2$, $H(PW_2)$, $H(PW^*)$ are not modified. After that, it verifies if the current password is valid by comparing $H(PW_2)$ with the current hashed password $H(PW)$ in its database. If matched, then it replaces the current hashed password with the new one. After that, the node which receives the message from the user may send a message to the user to inform the password change process is successful.

V. Security Analysis

In this section, we analyze security of TTUA. TTUA can defend against typical attack on sensor network authentication as presented in [1]-[4] by including node compromised attacks, replay attacks, impersonate attacks, and DoS. In the following, we discuss how TTUA can defend against these attacks.

- **Secure against node compromise:** In our scheme, $CHs$ are equipped with tamper-resistant hardware, so they cannot be compromised. The keys and user information stored on $CHs$ cannot be disclosed to anyone. Therefore, the adversary cannot make use of the $CH$ to convince the sensor $s$ that she is a legitimate user. If the adversary compromise $s$, what she can get is a pair-wise shared key $K_{A,s}$ with the $CH$. However, it is not possible for her to use $K_{A,s}$ to authenticate herself to the sensor network because the key is shared between $CH$ and $s$ only.

- **Secure against replay attacks:** In TTUA, an adversary cannot use the previous message to login successfully. In the authentication phase, the password and a timestamp are hashed before being sent to the $CH$. If the adversary intercepts the message $(UID_1, H(H(PW_1) \oplus T_0), T_0)$ and reuses it somewhere else, the $CH$ can detect by checking the timestamp $T_0$. If $T_0 < T_{now}$, the $CH$ knows that the message has already been used. Thus, replay attacks are not possible.

- **Secure against impersonate attacks:** The proposed scheme can resist against impersonate attacks. That is he cannot impersonate legitimate user even he intercepts any message on communication. There are three cases as follows:

  - The adversary intercepts the message $(UID_1, H(H(PW_1) \oplus T_0), T_0)$ between the user and the $CH$, and attempts to change $UID_1$ with her own $UID^*$. The $CH$ can easily detects because $UID^*$ does not exist in its database.

  - With the above message, the adversary attempts to get the password from $H(H(PW_1) \oplus T_0)$. However, it is considered practically impossible for her to derive the password from the hashed value.

  - The adversary intercepts the message $UID_1, ID_A, T_0, MAC(K_{A,s}, UID), ID_A || T_0)$ between the $CH$ and the sensor $s$. She then attempts to modify $UID$ in the message by her own $UID^*$, However, $s$ can detect this by building the MAC from $UID^*, ID_A, T_0$ and comparing with the received one. It is obvious $MAC(K_{A,s}, UID || ID_A || T_0) \neq MAC(K_{A,s}, UID^* || ID_A || T_0)$. Therefore, it is not possible for the adversary to impersonate a legitimate user.

- **Secure against Denial-of-Service (DoS) attacks:** There are two cases for an adversary to attempt DoS attacks to the sensor network:

  - She can use a powerful device such as a laptop to send many authentication requests to the sensor network. TTUA can mitigate the attacks by first checking if user ID exists in the $CH$’s database. If no, then it will not compute the hashing value of the password.

  - If she intercepts the broadcasting message between the BS and the $CH$ $(UID, H(PW), MAC(K_i, UID || H(PW)))$, she may get user id and the hashed password. She then instantly creates $UID, H(H(PW)) || T_0, T_0$ and requests authentication to $CHs$. In this case, the $CH$ has to instantly compute hash values to verify the request. If the $CH$ cannot handle such request storms, DoS is possible. To solve this problem, we set a time-out period for requests of each user ID. If the adversary sends a large amount of requests with the same user ID within a short period of time, the $CH$ will not accept and process those requests. Therefore, DoS is not possible at the $CH$. On the other hand, since the request is not passed at the $CH$, the $CH$ would not build a MAC to send to the sensor $s$. As a consequence, DoS is not possible at the sensor $s$.

VI. Performance Evaluation

To the best of our knowledge, TTUA is the first user authentication scheme for HSNs. Therefore, it is not possible to compare it with another scheme for HSNs. Instead, we compare TTUA with an existing user authentication scheme for homogeneous sensor networks in order to show the advantages of TTUA and the benefit of using HSNs. We select TJY [7] as it is one of the most efficient schemes that we have surveyed. We start with an analysis-based evaluation, and then simulation-based evaluation.

A. Analysis

We analyze performance of TTUA in terms of the storage requirement, communication overhead, and computation cost.

- **Storage:** In TTUA, the cluster head $CH$ has to store user IDs and hashed password values. Suppose there are $n$ users, user ID size is 8 bytes, hashed password value is 20-byte long. Thus, $CH$ has to store $28n$ bytes. In addition, each $CH$ has to store secret keys $\{K_{A,s}\}$. The probability of $CHs$ in a certain area must be sufficient so that the whole area is covered. This problem was well-investigated in [11] based on Vapnik-Chervonenkis (VC) theorem. So
### Table I
Comparison of memory storage (for 10 users)

<table>
<thead>
<tr>
<th>Node type</th>
<th>TTUA</th>
<th>TJY</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH</td>
<td>400 bytes</td>
<td>0</td>
</tr>
<tr>
<td>s</td>
<td>0</td>
<td>100 bytes</td>
</tr>
</tbody>
</table>

### Table II
Comparison of computational cost

<table>
<thead>
<tr>
<th>Phase</th>
<th>Node</th>
<th>TTUA</th>
<th>TJY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registration</td>
<td>CH</td>
<td>$T_E + T_H$</td>
<td>0</td>
</tr>
<tr>
<td>Authentication</td>
<td>CH</td>
<td>$T_H + 3T_{MAC}$</td>
<td>$2T_H + 2T_{XOR}$</td>
</tr>
<tr>
<td>total</td>
<td>CH</td>
<td>$2T_H + 3T_{MAC} + T_E$</td>
<td>$3T_H + 2T_{XOR}$</td>
</tr>
</tbody>
</table>

There are approximately 15 sensors \{s\} connecting to one CH. Each secret key is 64 bits (8 bytes) long. Therefore, each CH stores $15 \times 8 = 120$ bytes of secret keys. Totally, each CH must store $28 \times n + 120$ bytes. In TJY, the user does not need to store any secret information, but each sensor has to store 8 bytes for a user ID and 2 bytes for a timestamp. Thus, it shall cost $10 \times n$ bytes of the storage. Assume $n=10$, the memory required is shown in Table I.

The CH such as iPAQ has 64 MB memory storage. If we assume the storage is reserved for maintaining user IDs and passwords only, then TTUA can maintain approximately 2,340 users. Meanwhile, the sensor (e.g., TelosB) has 1 MB memory storage. Thus, TJY scheme can maintain 102 users only. This means that if the memory storage is only used for storing user's information, TTUA can maintain 2,340 users, while TJY can only maintain 102 users (Fig. 3).

- **Computation:** We define $T_{MAC}$, $T_H$, $T_E$, and $T_{XOR}$ are computation cost of performing message authentication code (CBC-MAC), hash function (SHA-1), symmetric encryption, and XOR operations, respectively. To make the comparison with TJY scheme fairly, we consider the computation cost on both CH and login-node s in TTUA; for TJY, we only take into account of the login-node, regardless the gateway-node because it is a very powerful node such as a PC. The computation cost are summarized in Table II.

We can see that our scheme requires an additional computation cost on CH ($2T_H + 3T_{MAC} + T_E$). However, this computation cost is very small compared with that of sensors because CH is a powerful sensor node. The total computational cost for the sensor s in our scheme is smaller ($1T_{MAC}$) than TJY ($3T_H + 2T_{XOR}$). According to practical results on real sensor devices [16], one MAC costs 7.1 ms, while a SHA-1 costs 3.5 ms. We summarize the computation cost through an numerical comparison in Fig. 4. The figure shows that our computation cost on sensors is less than TJY scheme.

- **Communication:** To calculate computation cost, we define a number of notations as follows (all of these are in number of hops).
  - $C_{broadcast}$: Communication cost for broadcasting user ID and password to all CHs
  - $C_{U-A}$: Communication cost between the user and the cluster head $A$
  - $C_{U-s}$: Communication cost between the user and the sensor $s$
  - $C_{A-s}$: Communication cost between $A$ and $s$
  - $C_{s-GW}$: Communication cost between $s$ and the gate-way node (in TJY)
  - $C_{U-GW}$: Communication cost between the user and the gate-way node (in TJY)

Note that $C_{U-s} = C_{U-A} + C_{A-s}$ because any message sent to $s$ must be relayed through its cluster head $A$. The communication cost is summarized in Table III.

In TTUA, it requires an extra communication cost for broadcasting user IDs and passwords to all CHs. This is insignificant cost for the powerful CHs. In authentication phase, we can see that TTUA only requires commu-
communication between the user and the sensor $s$ ($2C_{U-s}$).
Meanwhile, in TJY, every authentication process requires extra communication with the gate-way node ($2C_{U-s} + 2C_{s-GW}$). Practically, how much the communication costs depending on physical distance between the login-node and the gate-way. The further distance is, the more it costs.

To illustrate this in a straightforward manner, we perform a number of experiments (see Section VI-B) to measure the communication cost. The user’s location and the login-sensor are randomly changed. The gate-way node is located at the center of the sensor field. The result is shown in Fig. 5. It shows that the communication cost of TTUA is about twice less than that of TJY.

B. Simulation

This section describes our simulation result and a comparison with TJY. The simulation result shows the average energy consumption and delay time of different network topologies. For each network topology, user’s location and the login-node are randomly changed within the sensor field.

a) Simulation model

We simulated the TTUA on SENSE simulator (Sensor Network Simulator and Emulator) [20]. For comparison, we also simulated TJY with the same network topologies and authentication scenarios.

The network deployment is similar to [11]. The default simulation testbed has 1 base station and 300 sensors randomly distributed in a 300m×300m area. There are additional 20 CHs in the sensor field [11]. The transmission range of a sensor $s$ and a CH is 60m and 150m, respectively. Sensors and CHs are formed in clusters. Each cluster has one CH. Sensors in the same cluster are connected with its CH via one or more hops. We use the same energy model used in ns-2.1b8a [21] that requires 0.66W, 0.39W, and 0.035W for transmitting, receiving, and idling, respectively. We set the power consumption rate according to [11][19] for SHA-1 and CBC-MAC calculation 0.48W. As analyzed in [17][18], we set the time consumption for computing a CBC-MAC and a SHA-1 is 7.1ms and 3.5ms, respectively. The simulation uses MAC802.11 Distributed Coordination Function (DCF). Two-ray ground [22] is used as the radio propagation model.

For routing in both TTUA and TJY, we applied Ad hoc On-Demand Distance Vector (AODV) protocol. User ID length is 8 bytes, SHA-1 value is 20 bytes. As discussed in [17], the choice of 4-byte MAC is not detrimental in the context of sensor networks. So we apply 4-byte CBC-MAC for every message.

We run five different network topologies. For each topology, five scenarios are applied, in which user’s location and the login-node are randomly selected. For TJY, we set the gate-way node in the center of the sensor field. We then average the results from those scenarios.

b) Results

Our simulation result is shown in Fig. 6 and Fig. 7. For one registration, the user authenticates 1, 5, 10, and 20 times.

The energy consumption of computation and communication in Fig. 6 shows that the energy consumption of TTUA is about twice less than that of TJY. This is consistent with our analysis result in Section VI-A. This is because our
computation cost is less than TTY. scheme and TTUA does not require any extra communication with the gate-way node during authentication process. In Fig.7, the delay time of TTUA is also much less than that of TTY. It means that TTUA responses to user authentication request more quickly than TTY. For both schemes, the delay time for the first authentication is relatively higher than later ones because the AODV routing protocol takes time to request a routing-path establishment.

Both performance analysis and simulation results have shown that TTUA met the requirements and is dominant over the existing scheme.

VII. CONCLUSION AND FUTURE WORK

This paper presents Two Tier User Authentication scheme (TTUA) for heterogeneous sensor networks. TTUA takes advantages of powerful sensor nodes to increase security and efficiency. Any user can use user ID and password to access any node in the sensor network in a real-time manner. Through security analysis, we show that TTUA is secure against node compromise, reply attacks, and forgery attacks. Both performance analysis and simulation results have shown that TTUA achieves better energy-efficiency compared with the existing scheme such as TTY. The proposed scheme is more secure and energy-efficient than existing approaches.

For our future work, we are going to implement the scheme on our real sensor devices. More importantly, we will provide a mutual authentication, in which sensors must authenticate to users.

REFERENCES

Asynchronous and distributed nonlinear regularization for wireless sensor networks

Nicolas Maréchal  
CEA, LETI, MINATEC  
Grenoble, France  
nicolas.marechal@cea.fr

Jean-Benoît Pierrot  
CEA, LETI, MINATEC  
Grenoble, France  
jean-benoit.pierrot@cea.fr

Jean-Marie Gorce  
CEI - INSA Lyon  
Lyon, France  
jean-marie.gorce@insa-lyon.fr

Abstract—In this paper we introduce an alternative to standard gossip averaging algorithms for wireless sensor networks. The proposed algorithm takes opportunity of the intrinsic broadcasting nature of the wireless medium and copes well with the unreliability of radio links. Indeed, this algorithm works asynchronously and in a distributed manner using one-way exchanges (acknowledgements are not required). It relies on the regularization formalism as used in image processing, and we show that it provides a tunable approximation of distributed averaging.

Index Terms—Distributed algorithm, sensor networks, networked systems, consensus algorithms, information fusion.

I. INTRODUCTION

Sensor networks are gaining much interest because of the great amount of applications they are candidates to. They are commonly used for monitoring physical phenomena on wide areas such as hygrometry, landslides or fires, but also for tracking purposes and in military warfare ([1],[2],[3]). Such a network consists in a large amount of small entities, the nodes, equipped with low cost hardware in order to balance the total network cost. The direct drawbacks of low cost hardware are numerous: severe energy constraints (battery lifetime), poor CPU and storage abilities, low transmission rates and small communication ranges. Facing these limitations and objectives, wireless sensor networks (WSN) have to self-organize their exchanges and the manner in which sensor nodes must achieve their mission. As in most cases, a centralized computation of a distributed quantity becomes intractable or inappropriate, robust distributed algorithms are needed. The biggest part of algorithm design for WSN is dedicated to improving the performances while preserving energy consumption. For example, several data fusion schemes developed in order to provide a good and compact representation of the observed phenomenon can be made on the basis of a high number of low quality measures [4] and simple local interactions between neighbour nodes (gossiping). The particular class of distributed consensus algorithms is of great interest: they provide a robust way of homogenizing parameters among network nodes [5]. More specifically, average consensus algorithms [6] seem to be a good choice whenever the stability and the quality of the consensus point is a critical issue, and extend to a wide panel of data fusion tools such as estimators for statistical moments, linear regression and polynomial map fitting ([4],[7]). These algorithms may also be used as building block for more practical applications such as clock synchronization [8]. However, the solution proposed in these references hides restrictive mechanisms under their apparent simplicity: local exchanges between neighbour nodes need to be bidirectional and reliable (acknowledged). Despite its inherent distributivity, this method doesn’t match well with two major characteristics of radio communications. First of all, radio communications are locally diffusing: several entities are able to receive the same information simultaneously, even if they are not supposed to. Secondly, they are subject to interference and noise: information should vanish in an unpredicted manner, and thus acknowledgement is an expensive technique which restrict the wireless nodes to communicate with their closest neighbors. In this paper, we propose a new family of algorithms based on regularization, which approximates only the convergence to the desired consensus but in turn fits perfectly with the above mentioned radio communication features.

Regularization is a concept widely used in image processing for denoising purposes ([9],[10]). It consists in a smoothing process which generally operates on each pixel according to its neighbours’ intensity. This process can be achieved by either a direct transformation, or a self-stabilizing iterative scheme. In many cases, regularization can be modelled as the minimization of a potential function that captures the dissimilarities between neighbour pixels. For WSN applications, pixels are replaced by nodes, intensity by measurements and/or states, and neighbouring relationships are induced by the network connectivity. Even if independently obtained, similar ideas have been recently proposed in [11] with different assumptions and purposes (see section 3 for a detailed comparison and comments). Nevertheless, the novelty of the approach presented in this article is to provide a fully asynchronous solution which uses efficiently the communication channel, while being easy to implement and is robust to data losses and topology changes. As seen in the third section, it also offers a way for trading between accuracy and computation time.

The article is organized as follows. In section II, the classical formalism of regularization is described, and an asynchronous solution fitting to the needs and constraints of WSN is proposed. The convergence of this algorithm is proved under relatively unrestrictive assumptions. In section III, the authors
explain how such a regularization using simple quadratic functions may provide a robust (but approximate) alternative to gossip-averaging algorithms for the problem of average consensus, and the worst-case quality of the approximation is derived. Simulation results for a simple and idealistic scenario are given in section IV to illustrate what can be obtained from the regularization algorithm both in finite time and asymptotically. A short conclusion and perspective are drawn in section V.

II. DISTRIBUTED OPTIMIZATION PROBLEM

The regularization process as intended in this work consists in minimizing a potential function that can be written as the sum of pairwise potentials (couplings) between:

- the current regularized states of two neighbour nodes.
- the current regularized state of a node and a static reference (local measurement, known global parameters...).

These couplings capture the discrepancy between geographically near states, and their minimization leads to a partial homogenization of the data while keeping coherence with observations. Some example of attachment to a global reference can be found in Tikhonov regularization [12] of ill-posed problems. For instance, one can introduce a penalty upon states amplitude by attaching it to the 0 value through a potential function called \( \psi_e \).

A. General formulation

In the following, \( x_i \in \mathbb{R} \) represents the regularized state of node \( i \), and \( z_l \in \mathbb{R} \) represents a local or global reference (measurement, parameter) which represents the prior information used for regularization. In image processing or in WSN, each pixel/node has generally a single attachment variable.

It is clear that such a situation should be avoided.

B. Connectivity constraints

Let us define a regularization graph \( G = (\mathcal{V}, \mathcal{E}) \), with \( \mathcal{V} = \{1, \ldots, n\} \). The states \( (i, j) \in \mathcal{E} \) \( \iff \beta_{ij} + \beta_{ji} \neq 0 \). In particular, we set \( \beta_{ij} \) to 0 whenever nodes \( i \) and \( j \) cannot communicate in a bidirectional manner. By bidirectional, it is not meant that any packet that \( i \) sends to \( j \) will be received, and respectively, but that the success probability of these transmissions are both not null. The reader should note that this condition is necessary to carry out the minimization locally. Now, we can uniquely partition \( G \) into maximal connected components \( G_k \). The potential function thus writes:

\[
U(X) = \sum_{k \in \mathcal{G}_k} \left\{ \sum_{l \in \mathcal{V}_k} \alpha_{il} \psi_e(x_i - z_l) + \sum_{i,j \in \mathcal{E}_k} \beta_{ij} \psi_n(x_i - x_j) \right\}
\]

Each \( U_k \) is strictly convex if and only if there is at least an attachment variable \( Z \) for this subgraph. Adding the fact that minimization variables are appearing in one and only one \( G_k \), one can clearly assert that the minimization of \( U \) can be made separately on each \( G_k \). Thus we now consider a connected weighted graph \( G \) with \( n \) nodes.

C. Synchronous Jacobi algorithm

In a centralized or synchronous optimization scheme, a unique vector \( X(t) \) is known, and its entries are updated concurrently according to a minimization strategy: Gauss-Seidel, conjugate gradient or Jacobi methods can be used for instance. We focus here on the Jacobi algorithm for its easiness to be distributed.

Definition 1 (Jacobi algorithm). Let \( U : \mathbb{R}^n \rightarrow \mathbb{R} \) be a function to minimize, and \( J_i : \mathbb{R}^n \rightarrow \mathbb{R} \) the operator defined by:

\[
J_i(x_1, \ldots, x_n) = \arg \min_{y_i \in \mathbb{R}} U(x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)
\]

(3)

Given a vector \( X(0) \), the synchronous Jacobi algorithm consists in iterating simultaneously each \( J_i \) on \( X(t) \), and storing the result as the \( i \)-th component of vector \( X(t+1) \):

\[
X(t+1) = [J_1(X(t)), \ldots, J_n(X(t))]
\]

(4)
In other words, each component is minimized separately, but they are all updated simultaneously. This principle should be understood as the minimization of a global objective function on the basis of simultaneous local and independent actions. Such a method is thus of great interest for systems requiring distributed computing, in particular wireless sensor networks. Let us point out that, whenever a node \( i \) applies the operator \( J_i \), it requires the knowledge of the state \( x_{ij} \) of every node \( j \) sharing with \( i \) a positive coupling weight \( \beta_{ij} \) or \( \beta_{ji} \); bidirectionality as understood in subsection II-B is thus necessary.

**D. Asynchronous optimization**

In a WSN, the nodes exchange some information but for time, space and energetic complexity reasons, no node has a complete knowledge about the actual state of all the other nodes. Let us assume that each node stores in a buffer the latest states it received from its neighbours. Having \( n \) nodes, there are \( n \) buffer vectors, each one storing a partial knowledge of the latest states of the nodes in the neighborhood of its owner. In the following, a node may have the choice to refresh its own state at the view of the neighbours’ states it buffered. Once updated, it may choose to transmit its new state to its neighbours.

Let \( J \) be an operator from \( E = E_1 \times \cdots \times E_n \) to \( E \), and \( H_i \) its projection on \( E_i \), i.e.:

\[
[y_1, \ldots, y_n] = J(X) \Rightarrow H_i(X) = [J(X)]_i = y_i
\]  

(5)

Pulling the definition of an asynchronous iteration given in [13] into the framework given in [14], chapter 6, one obtains the following formulation:

**Definition 2 (Asynchronous iteration [14]).** For \( t \in \mathbb{N}, \) let:

- \( \mathcal{I}(t) \subseteq \{1, \ldots, n\} \)
- \( \tau_i^j(t) \in \mathbb{N}, \forall i, j \in \{1, n\} \)
- \( T^t = \{ t \in \mathbb{N} : i \in \mathcal{I}(t) \} \)

such that:

1) \( \tau_i^j(t) \leq t \) for \( i \in \{1, \ldots, n\}, t \in \mathbb{N} \)
2) \( \lim_{i \to \infty} \tau_i^j(t) = \infty \) for \( i, j \in \{1, \ldots, n\} \)
3) \( \text{Card}(T^t) = \infty \) for \( i \in \{1, \ldots, n\} \)

**Given an initial guess** \( x^0 \in E = E_1 \times \cdots \times E_n, \) **the iteration**

\[
x_i(t + 1) = \begin{cases} 
J_i(x_1(\tau_1^i(t)), \ldots, x_n(\tau_n^i(t))) & \text{for } i \in \mathcal{I}(t) \\
\delta_i(t) & \text{for } i \notin \mathcal{I}(t)
\end{cases}
\]

is termed an asynchronous iteration (\( \mathcal{I}(t) \) then indicates which components are updated at time/iteration \( t \)).

In Definition 2, the variable \( \tau_i^j(t) \) should be interpreted as the last date, up to time instant \( t \), at which node \( j \) has seen the state variable of node \( i \). These assumptions allow WSNs supporting packet losses and transmission delays: the situation in which a packet sent by node \( i \) is lost at node \( j \), where it should be taken in consideration, falls in this framework as it just decreases \( \tau_i^j(t) \).

**Proposition 1.** Under the assumptions of definition (2) and assumptions 1, the asynchronous minimization of \( U(X) \) using the Jacobi algorithm converges to the unique minimizer \( X_{opt} \) of \( U \) for any initialization of the states variables, if \( \forall i, \exists l, \alpha_{il} > 0 \).

**Proof:** The proof of this proposition relies mainly on results and facts from [14]. In this book, section 6.3/p.434, the authors explain that an asynchronous iteration scheme applied to a function \( F \) converges to a fixed point \( x^* \) of \( F \) if this function is a contraction mapping with respect to a weighted maximum norm. In this paper, \( F \) is the Jacobi operator applied to the potential function \( U \). For a similar case, the authors of [14] proved in proposition 3.10, p.221, that the contraction property of \( F \) relies directly on \( R(X) := X - \nabla U(X) \) which has to be itself a contraction mapping w.r.t. the same weighted maximum norm. Now, the contraction property of \( R(X) \) may be proved using proposition 1.11 (p.194). A direct calculation ensures that the hypotheses of this last proposition are verified under assumptions 1 when there is at least an attachment variable for each regularization state. The reader should note that, as \( U \) has a positive definite Hessian matrix \( (\nabla^2 U(X)) \) is nothing but the sum of a diagonal matrix with positive elements and a weighted graph Laplacian matrix, and hence \( \forall X, \nabla^2 U(X) \) is clearly positive definite), it is strictly convex and assumed coercive by assumptions 1: a classical result of nonlinear optimization is that \( U \) admits a unique minimum/minimizer \( X_{opt} \). A technical point appearing in hypotheses b) and c) of proposition 1.11 (p.194) has to be verified in order to finish the proof. For this, we need to verify the boundedness of the diagonal terms of \( \nabla^2 U \) by some positive \( K \) in b) and the positivity of the gap in c). These limitations are clearly verified at each \( X \) of a closed cube \( \Omega^n \) (compact), where \( \Omega \) is the minimal closed interval of \( \mathbb{R} \) containing every \( z_i \) and initial values \( x_i(0) \); the Heine-Borel theorem conclude on the existence of \( K \) and \( \beta \).

The convergence rate of this kind of algorithm is an still open research question. However, the case of quadratic coupling function is discussed in section III.

**Remark 1.** If the coupling functions do not verify assumptions 1.3 and 1.4, non-convexity of the potential function or non-existence of a global minimizer may occur. Relaxing these restrictive assumptions would allow the use of the results from the theory of image segmentation ([9],[15],[10]) for WSN, thus opening very interesting opportunity for natural clustering, partial synchronization, etc. At this time, we kept these issues out of the scope of this paper.

**Remark 2.** Sporadic topology changes (displacement, appari-
chance and crashes) are naturally supported by the previous approach. To cope with such situation, an easy solution comes as every node owns a table containing the last states it received from its neighbours: a node that detects a topology change (detection of an unreferenced node, ...) just adds (resp. removes) the entries corresponding to the new nodes (resp. the nodes that vanished) in its vicinity. The assumptions of proposition 1 still ensure convergence since the initial guess
for neighbours’ states does not matter. However, the objective \(X_{opt}\) changes with topology.

### III. ON THE USE OF REGULARIZATION AS AN APPROXIMATION FOR AVERAGE CONSENSUS

#### A. Principle and error bound

As explained in the introduction, gossip-based average consensus algorithms are good candidates for computing aggregates in sensor networks. The main objective they supply is the computation of average values on the basis of asynchronous and local interaction between neighbour nodes. In the standard version of this algorithm, a node that decides to initiate an interaction, say \(i\), chooses a neighbour according to a policy (usually at random), say \(j\). During the interaction, \(i\) and \(j\) must exchange and average their states:

\[
x_i(t+1) = x_j(t+1) = \frac{x_i(t) + x_j(t)}{2}
\]

(6)

Given an initial state \(x_i(0) = z_i\) on each node, each node asymptotically obtains the value \(z_{avg}\) defined by:

\[
z_{avg} = \frac{1}{n} \sum_{i=1}^{n} z_i
\]

(7)

For this purpose, each node gets a single-position buffer which stores a state variable. These state variables are exchanged between neighbour nodes and processed locally by means of linear transformations whose complexity depends on the size of the interacting community at each iteration. Despite its apparent simplicity, this formalism hides a penalizing mechanism: acknowledgement of state exchanges (see [16]). By verifying that state variables were received by all candidates to the interaction, it ensures (almost surely) that the algorithm converges to the desired average, i.e. \(z_{avg}\). If acknowledgement were removed, the work published in [17] may be used to prove the convergence to a consensus: the algorithm generates a random walk which stabilizes at a consensus value that may be different/far from the desired average \(z_{avg}\). Moreover, the implementation of a protocol for gossip averaging algorithms using radio communication channels becomes quite challenging if nodes access to the medium in a concurrent manner (e.g. ALOHA or slotted-ALOHA MAC protocols): even the implementation provided in [16] may suffer from deadlock issues if timing between packet exchanges is not correctly managed. In many applications, the “averaging property” is crucial ([18], [4]) and ensures the coherence of computed aggregates. Facing these needs and limitations, the regularization scheme introduced in this paper provides a robust alternative to gossip-based average consensus algorithms for WSN applications. The price in this paper provides a robust alternative to gossip-based average consensus algorithms for WSN applications. The price

The “averaging property” is crucial ([18], [4]) and ensures the coherence of computed aggregates. Facing these needs and limitations, the regularization scheme introduced in this paper provides a robust alternative to gossip-based average consensus algorithms for WSN applications. The price of robustness lies is the fact that regularization only tends to an approximation of the average. Fortunately, controlling the fineness of the approximation is possible as proved below.

In the following, we denote \(\bar{Z}\) the \(n\)-dimensional vector of which components are equal to \(z_{avg}\). Now, fix:

- \(\alpha_{ij} = \delta_{ij}/2\), i.e. \(x_i\) is only attached to the local measure made by node \(i\).
- \(\beta_{ij} = \beta/4\) if \(i\) and \(j\) are neighbours \((i \sim j)\), 0 otherwise.
- \(\psi_n(x) = \psi_n(x) = x^2\)

In [11], a similar quadratic minimization is suggested, but its processing is assumed to be synchronous and the assumptions on communications are quite different. For instance, communication noise is introduced but packet losses are ignored. If communication noise is removed, the approach presented here may be seen as a superset of [11] in the sense that full asynchrony in local state processing and transmission becomes possible. This gain is considerable since the energy spent for ensuring synchronization of a large scale WSNs may have be highly non-negligeable. Moreover, the main purpose of [11] is to provide a solution robust to communication noise, while a second aim is reached in our algorithm, that is the efficient use of the diffuse nature of wireless communications, in comparison to standard gossip averaging algorithms.

As we consider quadratic coupling functions, a direct calculation shows that the Jacobi operator at node \(i\), i.e. \(J_i\), is given by:

\[
J_i(x_1, \ldots, x_n) = \frac{z_i + \beta \sum_{j \sim i} x_j}{1 + \beta N_i}
\]

(8)

where \(N_i\) is the number of neighbours of node \(i\). The equilibrium vector \(X_{opt}\), i.e. the unique vector such that \(J(X_{opt}) = X_{opt}\), also satisfies the following equation:

\[
X_{opt} = M^{-1} Z, \quad M = I + \beta L_G, \quad Z = [z_1, \ldots, z_n]^T
\]

(9)

where \(L_G\) is the standard graph Laplacian of \(G\) (see [19]). Thus, \(X_{opt}\) depends inherently on the network topology. However, the next statement proves that the impact of the topology can be lowered as much as desired:

**Proposition 2.**

\[
\|X_{opt} - \bar{Z}\|_2 \leq \frac{1}{1 + \alpha(G)\beta} \|Z - \bar{Z}\|_2
\]

(10)

where \(\alpha(G)\) is the algebraic connectivity of \(G\), i.e. the second smallest eigenvalue of its Laplacian.

**Proof:** As \(G\) is connected, the eigenvalues of \(L_G\) verify \(0 = \lambda_1 < \lambda_2 = \alpha(G) < \cdots < \lambda_n\) (see [19]). Moreover, the inverse of \(M\) has the same eigenvectors as \(M\) and eigenvalues \(\lambda'_i\) given by:

\[
\lambda'_i = \frac{1}{1 + \beta \lambda_i}
\]

(11)

The vector \(\mathbf{1} \triangleq [1/\sqrt{n}, \ldots, 1/\sqrt{n}]^T\) is the only eigenvector associated:

- with \(\lambda_1 = 0\) for \(L_G\)
- with \(\lambda'_1 = 1\) for \(M^{-1}\)

Thus, \(Z = z_{avg} \mathbf{1}\) is left unchanged by \(M^{-1}\). As a consequence, the initial averaging error vector \(Z - \bar{Z}\) is orthogonal to \(\mathbf{1}\): as proved in the following, this vector is the part of \(Z\) that will be affected by the regularization factor \(\beta\). From the equality

\[
X_{opt} - \bar{Z} = M^{-1}(Z - \bar{Z})
\]

(12)
one gets the following important result that ends the proof of the proposition:
\[ \|X_{\text{opt}} - \bar{Z}\|_2 \leq \lambda_2 \|Z - \bar{Z}\|_2 = \frac{1}{1 + \beta \Delta} \|Z - \bar{Z}\|_2 \quad (13) \]

As a consequence, the error between regularization and averaging only depends on the \( \beta \) parameter, and the algebraic connectivity \( \alpha(G) \) of the network (see [20], [19]...). In particular, one should keep in mind that for any connected graph, the following bound holds [19]:
\[ \alpha(G) \geq 2(1 - \cos(\pi/n)) \quad (14) \]
This induces a loose upper bound for the asymptotic error norm.

B. Convergence rate

The previous result indicates that setting \( \beta = \infty \) would result in a perfect average consensus. However, in this case the impact of attachment variables vanishes and the algorithm becomes unstable: convergence is not ensured, there exists a hyperplane of equilibrium points, and their stability is not guaranteed. In other words, if the algorithm converges to an equilibrium state, it may be far away from the average consensus.

Under the general assumptions of this article, the convergence is ensured but the question of its exact rate is still open. In the case of linear interactions, corresponding to the use of quadratic coupling functions, some results in literature suggest a geometric convergence rate in two important cases, that is:
• when the algorithm is synchronous, i.e. every node applies simultaneously the Jacobi operator and transmit its state correctly to its neighbours [21].
• when the delays \( t - \tau_i^j(t) \) are bounded (see [14]).

For instance, the synchronous counterpart of the algorithm presented in this paper converges geometrically (see [21]) at a rate given by:
\[ \rho \left( (I + \beta \Delta)^{-1} A \right) \quad (15) \]
where \( \Delta \) is the standard diagonal degree matrix of \( G \), \( A \) its adjacency matrix, and \( \rho(\cdot) \) denotes the spectral radius. It becomes trivial that even the convergence rate of the synchronous version tends to 1 as \( \beta \) grows up to \( +\infty \), and thus the algorithm slows down drastically.

Some tracks around the case of probabilistic delays in linear interactions (corresponding to quadratic coupling functions) are given in [22], and also suggest a geometric convergence rate.

In the following, we give a proof that the convergence rate is also of geometric type under the assumption of a stationary communication process:

**Theorem 1.** Let \( S(t) \) be the set of active links at time \( t \), i.e. the set of successful unidirectional packet transmissions. If \( S(t) \) is a random process satisfying the following assumptions:

1) the random variables \( S(\cdot) \) are i.i.d

2) \( \forall t, \) each link of the connectivity graph belongs to at least one of the possible outcomes of \( S(t) \), having non-zero probability

then the algorithm converges geometrically, i.e. there exists a constant \( 0 < \rho < 1 \) such that:
\[ \|X(t) - X_{\text{opt}}\| = O(\rho^t) \quad \text{almost surely} \]
i.e. with probability one.

**Proof:** In the following, \( \| \| \) will denote the Euclidean norm for vectors and its induced norm for matrices, i.e. the spectral norm (see [23] for details). First, let us remark that the algorithm may be formulated as a vectorial system:
\[ \bar{X}(t + 1) = S(t).J_1 \bar{X}(t) + J_2 \bar{Z} \quad (16) \]

In this update equation, \( \bar{X} \) is a vector containing for each node:
• its own state
• the last state it has seen of each of its neighbours

This vector is thus obtained by stacking the buffer of each node. \( J_1 \) and \( J_2 \) are matricial operators corresponding to the Jacobi method for minimization, and \( S(t) \) is a transmission matrix which describes successful packet transmissions. It is possible to construct a vector \( \bar{X}_{\text{opt}} \) from \( X_{\text{opt}} \) by the following procedure: if the \( k \)-entry of \( \bar{X} \) corresponds to the state of node \( j \) as seen by node \( i \), then the \( k \)-th entry of \( \bar{X}_{\text{opt}} \) is set to \( x_{j_{\text{opt}}}^k \), i.e. the \( j \)-th entry of \( X_{\text{opt}} \). A direct calculation shows that \( \bar{X}_{\text{opt}} \) is a fixed point of equation (16). Thus, one obtains:
\[ \bar{X}(t + 1) - \bar{X}_{\text{opt}} \]
\[ = S(t).J_1 \bar{X}(t) + J_2 \bar{Z} - (S(t).J_1 \bar{X}_{\text{opt}} + J_2 \bar{Z}) \]
\[ = S(t).J_1 (\bar{X}(t) - \bar{X}_{\text{opt}}) \quad (17) \]
\[ = S(t).J_1 (\bar{X}(t) - \bar{X}(t)) \quad (18) \]
\[ = S(t).J_1 (\bar{X}(t) - \bar{X}(t)) \quad (19) \]

If the local states of nodes are converging to their optimal value, \( \bar{X}(t) \) must converge to \( \bar{X}_{\text{opt}} \). Thus, there is an equivalence between the convergence of local states of the asynchronous iteration, and the convergence of equation (16). By introducing an orthonormal basis of initial vectors, and using proposition 1 and the assumptions of definition 2, it is possible to show that:
\[ \forall \varepsilon > 0, \exists T \in \mathbb{N}, \forall \bar{X}(0) > 0 \text{ s.t.} \quad \|\bar{X}(0)\| = 1, \]
\[ t \geq T \Rightarrow \|\bar{X}_{\text{opt}} - \bar{X}(t)\| < \varepsilon \quad (20) \]

In other words, one can find a universal time \( T \) such that the residual error \( \|\bar{X}_{\text{opt}} - \bar{X}(t)\| \) is arbitrarily small for any unit vector \( \bar{X}(0) \). This universality plays an important role in the further development. Let \( \Phi(t) \) denote the matrix defined by
\[ \Phi(t) \triangleq S(t).J_1 S(t-1).J_1 \ldots S(1).J_1 S(0).J_1 \quad (21) \]

This matrix is obtained as the product of elements of a random sequences of matrices \( S(\cdot).J_1 \), all being i.i.d. random variables built upon the process \( S(t) \). Thus, one may form the probability space \( (\Omega, \mathcal{F}, \mathbb{P}) \) of such sequences (see [24] for background on modern probability theory). The reader should
note that the set of sequences of transmission matrices that do not verify the assumptions of 2 has measure 0 w.r.t. the probability measure $\mathbb{P}$. Now, the following theorem can be used:

**Theorem 2** ([25],[26],[27]). Let $(B_k : k \geq 0)$ be a stationary ergodic sequence of random matrices for which $\mathbb{E}[\log \max(\|B_0\|, 1)] < \infty$. Then, there exists a deterministic constant $\gamma$ (known as the Lyapunov exponent) such that

$$
\frac{1}{k} \log \|B_k \ldots B_1\| \to \gamma \quad \text{as } k \to \infty.
$$

Furthermore

$$
\gamma = \lim_{k \to \infty} \frac{1}{k} \mathbb{E}[\log(B_k \ldots B_1)]
$$

and

$$
\gamma = \inf_{k \geq 1} \frac{1}{k} \mathbb{E}[\log(B_k \ldots B_1)]
$$

The reader should note the following remarks:

1) The limit $\gamma$ does not depend on the realization of the random sequence.

2) Almost surely means that the set of sequences satisfying the property has probability one, i.e. the probability of the converse is zero.

3) As the matrix norms are topologically equivalent, the definition and value of $\gamma$ do not depend on the chosen norm.

4) Depending on the author, the matrix product is taken from the left to the right or conversely: the statement still holds when the orientation is fixed.

As the set of sequences of transmission matrices satisfying the assumptions of proposition 1, and the set of sequences satisfying theorem 2 both have probability one, their intersection has probability one too. Let us consider that $\Phi(t)$ is obtained from a sequence of this intersection. From theorem 2, $\gamma \geq 0$ would imply that $\forall t, \|\Phi(t)\| \geq 1$. Thus, $\forall t$, there would exist a unit vector $\tilde{V}_i$ such that $\|\Phi(t)\tilde{V}_i\| \geq \|\tilde{V}_i\| = 1$. This clearly contradicts the convergence, and then, $\gamma < 0$. From equation (22), one gets:

$$
\frac{1}{t} \log \|\Phi(t)\| = \gamma + o(1)
$$

and then:

$$
\|\Phi(t)\| = (e^{\gamma + o(1)})^t
$$

For sufficiently large $t$, the term $\gamma + o(1)$ is necessarily negative, and may be upper bounded by a constant $0 < \rho < 1$ such that:

$$
\|\hat{X}^{opt} - \tilde{X}(t)\| = \|\Phi(t)(\hat{X}^{opt} - \tilde{X}(0))\| = O(\rho^t)
$$

**IV. SIMULATIONS**

**A. Conditions**

This part is dedicated to simulation results, obtained with Matlab, that illustrate the behavior of the regularization algorithm. A network of $n = 400$ nodes is considered, each being uniformly distributed at random on a square $S$ of width 200 meters. In order to compare performances under realistic channel conditions, the probability of successful packet transmission between to nodes separated by a distance $d$ is given by the model from [28], i.e.:

$$
p_{suc}(d) = \begin{cases} 
1 - (d/R)^{2\alpha}/2 & \text{if } 0 \leq d \leq R \\
(2 - d/R)^{2\alpha}/2 & \text{if } R < d \leq 2R \\
0 & \text{otherwise}
\end{cases}
$$

where $R$ is a reference distance with success probability $p_{suc} = 0.5$, $\alpha$ is parameter related to fading, and $d$ the distance of communication. Without lack of genericity, we set $\alpha = 2$, $R = 20$ m, and two nodes are neighbours if their distance is smaller than $R$. Nodes are observing a spatial phenomenon depicted on figure 1, which is a combination of a sine wave, a Gaussian surface and an additive noise field. These measurements are denoted by $h(s_i)$ where $s_i$ stands for the location of node $i$ on $S$. Having the topology and the phenomenon fixed, the only remaining variables are the regularization parameters. For this purpose, the model from section III is used:

- $\psi_e(x) = \psi_n(x) = x^2$
- $\beta_{ij} = \beta/4$ if $i \sim j$, 0 otherwise
- $\alpha_{ij} = 1/2$ if $i = j$, 0 otherwise

In other words, each node has a single attachment variable corresponding to its own observation. The value of $\beta$ is successively taken from the set $\{0, 1, 10, 100\}$ to illustrate the impact of its magnitude on convergence speed and accuracy. Moreover, the regularization algorithm will be compared with a gossip averaging algorithm consisting of double acknowledgement process (communication must be initiated, then acknowledged by the destination and then confirmed by the initiator). For both algorithms, an iteration consists in choosing a node uniformly at random (see [29] for details): it only transmits its current state in the case of the regularization, and initiates a complete interaction with a random neighbour in the case of gossip averaging.

**B. Results**

This section begins by some results about the quality of the optimal state vector $X^{opt}$ with respect to average consensus vector $\bar{Z}$. Two error norms $\|X^{opt} - \bar{Z}\|_2$ and $\|X^{opt} - \bar{Z}\|_\infty$, together with the bound provided by equation (13) are plotted on figure 2 for some values of $\beta$ ranging from 0.1 to $10^4$. As predicted in section III, the accuracy of the approximation $X^{opt}$ increases with $\beta$, and the residual error can be made arbitrarily small by increasing $\beta$.

After considering the asymptotic accuracy of the proposed algorithm, the attention is now focused on the dynamical behavior of both regularization and gossip averaging. The state-to-average mean error norm $\mathbb{E}[\|X(t) - \bar{Z}\|_2]$ and the state-to-optimum mean error norm $\mathbb{E}[\|X(t) - X^{opt}\|_2]$ are respectively plotted on figures 4 and 3 as functions of the number of exchanged packets in order to compare the energy consumption of each algorithm. The error norms are computed by averaging over 1000 realizations. As predicted by theory,
regularized states converge to the optimal ones, and gossip averaging reaches the average consensus. In the case of regularization, these states do not tend to the average consensus but stay in a neighborhood, the diameter of which should be estimated on figure 2. Figures 4 and 3 also illustrate that the regularization scheme may perform much faster than the average consensus algorithm for small $\beta$ and attains rapidly its approximation of the average consensus. Thus, the need of acknowledgements for a gossip averaging algorithm (see [16]) should drastically reduce its performances in comparison to the regularization in presence of packet losses. In addition to that, the authors are confident into the possibility of speeding up the convergence by smartly increasing the $\beta$ parameter.

V. CONCLUSION

In this paper, we proposed a new algorithmic approach for solving the average consensus problem in an approximate but robust manner. The provided approach has two main advantages that are to be massively distributed and asynchronous, and intrinsically tolerant to data losses. These characteristics make the regularization algorithm adapted for systems implying nodes communicating over a wireless and/or diffusing medium. Just as standard gossip average consensus algorithms, the computable quantities can be directly extended to the computation of other aggregates like sample variance or linear LMS fitting. Moreover, the authors are confident in the optimization potential of non-quadratic coupling functions for addressing clustered versions of consensus algorithms, and will consider this issue in a further article.
A Distributed Constant Factor Self-Protection Algorithm for Wireless Sensor Networks

Kamrul Islam
School of Computing
Queen’s University
Kingston, Canada
Email: islam@cs.queensu.ca

Selim G. Akl
School of Computing
Queen’s University
Kingston, Canada
Email: akl@cs.queensu.ca

Abstract

Wireless sensor networks have drawn a lot of attention owing to their potential use in different surveillance applications including object monitoring, path protection, area coverage, and so on. Since sensors provide monitoring to other objects, it is often necessary to give them a level of protection such that sensors can take certain actions when attacks are targeted on them. One natural idea is to monitor sensors by their neighbors such that the neighbors can inform the base station when monitoring sensors are in danger. Keeping this in mind, the $p$-self-protection subset problem has been formally introduced in [4] to find a subset of sensors such that for any sensor in the network there are at least $p$ active sensors from the subset that monitor it.

The authors in [4] first show that finding minimum 1-self-protection subset is NP-complete and then give an $O(\log n)$-approximation algorithm. Later, with a view to obtaining a constant factor approximation ratio, centralized and distributed algorithms to solve the general $p$-self-protection subset problem are presented in [3]. In this paper, we present a counterexample showing that the algorithm in [3] fails to find a $p$-self-protection subset in the network. Later, we present a distributed algorithm for the $p$-self-protection subset problem and prove that the algorithm achieves a constant factor approximation ratio. To the best of our knowledge, this is the first constant factor approximation ratio to the $p$-self-protection subset problem. Our distributed algorithm runs in $O(\log^* n)$ time where $n$ is the total number of sensors in the network.

**Keywords:** Wireless Sensor Network, Self-Protection, Distributed Algorithm, Maximal Independent Set.

1. Introduction

Once deployed with a limited amount of energy, nodes in sensor networks continue to spend energy by sensing, gathering, and distributing data from the environment. Among myriad of applications, sensor networks are used for protection and surveillance of many different applications such as museum monitoring, military surveillance, object tracking, and so on. More specifically, nodes carry out protection and monitoring activities by collecting and sending timely signals to a base station to take appropriate measures in case some unusual events occur to the objects they monitor.

Although sensors are mainly used for protection of objects as mentioned, there are situations when it is vital to monitor the sensors as well. The issue of monitoring sensors arises because battery-powered sensors are vulnerable to damage, failure and attacks. If sensors themselves are incapable of performing monitoring, then how could we protect our objects? It is therefore important to ensure that sensors are active and functioning as desired. One of the ways this can be done is to require neighboring sensors to monitor the conditions of each other.

Imagine a scenario where among the set of sensors there is a single sensor which is monitoring a certain object. If, for some reason, this sensor malfunctions or runs out of energy then the object will be left unprotected. However, if there were some other neighboring sensors which could monitor this particular sensor then any status of that sensor (whether running out of energy or in danger) could be relayed by them to the base station so that proper action could be taken, such as, for example, replacing the sensor by other sensors. This scenario raises the issue that sensors themselves require a certain level of protection. The problem of protecting the sensors by other sensors is called self-protection. In its simplest form, a sensor network is self-protected if all sensors are protected by at least one active sensor (other than itself), where active sensors are sensors which monitor their neighbors by being active. In general, a wireless sensor network is $p$-self-protected if at any moment for any sensor there are at least $p$ active sensors (other than itself) that can monitor it. This problem has been formally introduced and investigated by Liu et al [4].

The rest of the paper is organized as follows. In Section 2 we describe related work and in Section 3 we provide definitions and assumptions that are used throughout the paper. A distributed algorithm (whereby the nodes in the network execute the algorithm individually) is presented in
Section 4. We provide a theoretical analysis of our algorithm in Section 5 followed by conclusions in Section 6.

2. Related Work

Since sensors are power-limited and monitoring requires sensors to be active all the time thereby consuming valuable energy, finding a small number of sensors to do the monitoring has been one of the main objectives of research in this regard. This small subset of active sensors are used to provide protection or coverage for every location of the area, in which they are deployed, and at the same time other sensors can be put in energy-efficient sleeping mode to conserve energy. A number of coverage problems, including protection of certain static objects with a required degree of fault-tolerance, and methods to solve them have been surveyed by Wu et. al [5], where the main focus is on the area coverage problem. A generalization of the area coverage problem called $k$-coverage, i.e., whether every point is covered by at least $k$ sensors, is studied by Kumar et al [10]. The $k$-coverage problem is directly related to the fault-tolerance issue in sensor networks. This is because $k$-coverage basically provides fault-tolerance in the network since failure of at most $k' < k$ monitoring nodes to monitor a particular area will not leave that area uncovered. Gui et al [7] propose algorithms for area coverage problems considering fault-tolerance, energy-consumption and quality of coverage.

A shift from the coverage problem to the problem of protecting the sensors themselves was first made by Liu et al [4]. In [4] they prove that finding minimum 1-self-protection subset (that is, finding a subset of nodes of minimum cardinality such that each node in the network is protected by at least one other node) is NP-complete through a reduction from the set-cover problem [11]. As a consequence they give a centralized algorithm for this problem and achieve a $2(1+\log n)$ approximation ratio, where $n$ is the total number of sensors. That is, their algorithm finds a 1-self-protection subset which is at most $O(\log n)$ times the cardinality of the minimum 1-self-protection subset. However, they do not give any algorithm for the general $p$-self-protection subset problem. The authors also show that focusing only on the quality of area or object covering does not necessarily make the sensors to be self-protected. Subsequently, a probabilistic definition of the self-protection subset (a protection is called $δ$-p-self-protection if the probability that a sensor is not p-self-protected is less than $δ$), as well as two randomized algorithms to solve the problem in a distributed fashion, are given in [4].

Later, the problem of finding minimum $p$-self-protection subset was thoroughly studied along with some variants such as requiring the protecting sensors be also connected, allowing sensors with varying sensing ranges, and so on, by Wang et. al [3]. In their paper, the authors claim that they have two constant factor approximation algorithms (in both centralized and distributed settings) for this problem. Unfortunately, their claim is in error. We provide a counterexample to show an instance of a graph where the $p$-self-protection algorithm does not form a subset that $p$-protects the nodes in $G$. They discuss other variants of the problem, namely, how $p$-self-protection can be achieved when the sensing radii of the sensors are not the same, or generating a certain number of $p$-self-protection subsets and activating them one after the other.

In this paper, we investigate the $p$-self-protection problem. By presenting a counterexample of the work of Wang et. al [3], we provide a distributed algorithm to solve the $p$-self-protection problem. To the best of our knowledge this is the first distributed constant factor approximation algorithm for the $p$-self-protection problem. This result is significant in two ways: first it achieves a constant factor which is better than the previous (logarithmic factor) results and second it is distributed.

3. Model and Definitions

We assume that sensors are deployed in the plane and model the network by an undirected unit disk graph $G = (V, E)$ where the vertex set $V$ denotes the set of sensors and $E$ represents the links $(u, v) \in E$ between two sensor nodes $u, v \in V$ if they are within their transmission range. We assume that every sensor node $u \in V$ has the same transmission range which is normalized to one. Define the neighbor sets $N(u)$ and $N[u]$ of sensor node $u$ as $N(u) = \{v | (u, v) \in E, u \neq v\}$ and $N[u] = N(u) \cup \{u\}$. By $N_k(u)$ we mean the set of nodes from $u$ which are at most $k$ hops away from $u$. For simplicity we use $N(u) = N_1(u)$. The degree of a node is the number of neighbors it has, that is, $|N(u)|$. Each node is identified by a unique index.

A set $I \subseteq V$ is called an independent set if no two nodes $u, v \in I$ are neighbors. A set $I \subseteq V$ is called an maximal independent set (MIS) if no independent set $T \supset I$ exists.

3.1. Minimum $p$-self-protection Subset

A sensor network is $p$-self-protected if, for any sensor, there are at least $p$ neighboring active sensors (a sensor is not a neighbor to itself) that monitor it. The minimum $p$-self-protection subset [3] is a subset $P$ of $V$ to be set as active sensors such that the network is $p$-self-protected and the cardinality of $P$ is minimized. See Figure 1 where the set of black nodes form an 1-self-protection (Figure 1 (a)) and a 2-self-protection (Figure 1 (b)) subsets. As mentioned before, it is proved in [4] that computing a minimum 1-self-protection subset is NP-complete, which implies that the general minimum $p$-self-protection subset is also NP-complete. From now on instead of using ‘$p$-self-protection’ we use ‘$p$-protection’. We provide a distributed constant
factor approximation algorithm for the \( p \)-protection subset problem when \( p \in \{1, 2, 3\} \), which, we think, are likely the more useful values of the protection parameter. This is because in real life if we allow more sensors \( p > 3 \) for the protection then we waste a lot of energy from the monitoring sensors which are doing nothing but providing protection to other sensors. Let \( OPT_1 \), \( OPT_2 \) and \( OPT_3 \) denote the minimum 1-, 2- and 3-protection subsets, respectively.

![Diagram](image)

Figure 1. Minimum 1-self-protection and 2-self-protection subsets.

We assume a synchronized message passing system as in [13], [14] in which time is divided into equal rounds. In each communication round a node of the network graph is capable of receiving messages sent in the previous round, performing local computations, and broadcasting a message to its neighbors. The time complexity of an algorithm is the number of rounds it needs to produce the desired results. It is well known that a synchronous message passing algorithm can be turned into an asynchronous algorithm with the same time complexity, but with a larger message complexity. Sensors have their own ids and connectivity information and obtain their neighbors’ indices and degrees, through exchanging “Hello” messages.

4. Minimum \( p \)-protection Subset Problem

In this section, we present a counterexample to show an instance where the \( p \)-protection algorithm fails to find a \( p \)-protection subset of nodes [3].


The algorithm in [3] for the general minimum \( p \)-protection subset problem is now described. In the first iteration, the algorithm [3] first finds an MIS \( M_1 \) based on the ranks (each node \( u \) is assigned a unique integer from \([1, n]\) called its rank, \( r(u) \)) of the nodes where a node in selected to be in \( M_1 \) if it has the largest rank among all its neighbors. Each node not selected in \( M_1 \) is assigned a unique rank \( r(v) = r(v) + k \ast n \), where initially \( k \) is assigned 1 and incremented by one at each iteration, and the ranks of the nodes belonging to the MIS are not changed. Then in the next iteration, the algorithm generates a new MIS \( M_2 \) from the nodes not selected in \( M_1 \) based on their ranks. Thereafter, \( k = k + 1 \) and ranks are updated by the equation \( r(v) = r(v) + k \ast n \) for those nodes which did not belong to \( M_1 \) and \( M_2 \). This procedure of generating new MISs is repeated \( p \) times, and it is claimed that this algorithm finds \( p \) MISs. Finally, as this algorithm continues, if a node \( u \in M_i \), \( 1 \leq i \leq p \) has fewer than \( p \) neighboring nodes in \( \bigcup_{i=1}^{p} M_i \), then \( u \) selects a neighbor \( v \) to protect \( u \) to such that \( u \) is \( p \)-protected. According to [3], the union of all \( M_i \)'s forms the \( p \)-protection subset.

In order to prove this claim (Theorem 2 in [3]), it is stated that for any node \( u \in \bigcup_{i=1}^{p} M_i \), \( u \) has at least \( p-1 \) protectors from \( \bigcup_{i=1}^{p} M_i \) since it has been protected by MIS nodes in every iteration except the iteration in which it is selected. Then if \( u \) has only \( p-1 \) neighboring nodes in \( \bigcup_{i=1}^{p} M_i \), it just adds an arbitrary neighbor \( v \) to protect itself.

However, this statement is not correct. Figure 2 shows a counterexample for the \( p \)-protection subset problem, where \( p = 4 \). Assume that the algorithm finds the first MIS \( M_1 \) consisting of nodes shown with label 1. According to the algorithm, another MIS \( M_2 \) is found consisting of the nodes that do not belong to \( M_1 \). However, in the example, there is no way to form a second MIS \( M_2 \) from the remaining nodes because any attempt to construct the second MIS without using nodes from the previous MIS will fail. More specifically, if we want to form the second, the third or the fourth MISs \( M_3 \) or \( M_4 \), each will consist of exactly two nodes (shown with label 2, label 3, and label 4, respectively) and none of which is an MIS of the whole graph. This means that in any case, either the middle label 1 node (when we form \( M_2 \) and \( M_3 \)) or the label 1 nodes on the left and on the right (for \( M_4 \)) are not protected by any sensors. Again, according to [3], for all nodes \( u \in \bigcup_{i=1}^{p} M_i \), \( u \) has at least \( p-1 \) protectors from \( \bigcup_{i=1}^{p} M_i \), given that \( u \) has already been protected by MIS nodes in every iteration except the iteration in which it is selected as a node in MIS. However, in the example of Figure 2, the middle label 1 node is not protected by \( p-1 = 3 \) nodes.

Finally, this erroneous observation leads them to claim that this can achieve a 10-approximation algorithm for the general \( p \)-protection subset problem. The authors of the paper [3] agreed that the counterexample we provide is correct [6].

![Diagram](image)

Figure 2. Counterexample to the proof of Theorem 2 [3].
4.2. Solution: 1-protection Subset Problem

For the minimum 1-protection subset problem we present a distributed version of the centralized algorithm in [3] that gives a constant approximation ratio. First, in order to prove a correct constant factor ratio we use a well-known property of MIS in unit disk graphs. Specifically we use the following fact from [9]. We reproduce their proof in order to facilitate the understanding of subsequent claims.

Theorem 4.1: [9] Let \( I \) be any MIS of the unit disk graph \( G \) and \( u \) be an arbitrary node in \( I \). The number of nodes in \( I \) that are exactly two hops away from \( u \) are at most 23.

Proof: The proof follows from the standard area argument. The disks of radius 0.5 centered at the nodes in \( I \) that are exactly two hops away from \( u \) all lie within the annulus centered at \( u \) of radii 0.5 and 2.5 and are disjoint. Therefore, the number of nodes in \( I \) that are exactly two hops away from \( u \) is less than \( \frac{4 \pi \cdot 2.5^2 - \pi \cdot 0.5^2}{\pi \cdot u} = 24 \).

Theorem 4.2: For the minimum 1-protection subset problem, the distributed algorithm finds a valid 1-protection subset \( A_1 \) and has size at most 33 times the minimum 1-protection subset \( OPT_1 \), that is, \( |A_1| \leq 33 \cdot |OPT_1| \).

Proof: Consider any node \( u \in V \). There are two cases, either \( u \in MIS \) or \( u \notin MIS \):

Case 1: If \( u \in MIS \) then there is no MIS node in its neighborhood but it can have at most 23 MIS nodes in (Theorem 4.1) its two-hop neighborhood. If each of these 23 nodes selects a unique node from \( N(u) \) and \( u \) selects its active neighbor then there will be at most 24 active nodes in \( N(u) \).

Case 2: If \( u \notin MIS \) then it can have at most five MIS nodes in \( N(u) \). Each of the five MIS nodes can arbitrarily select a unique node \( N(u) \) and at the same time each of 23 MIS nodes in the two-hop neighborhood can select a node in \( N(u) \). The MIS nodes and their selected neighbors are set active. Thus we have 33 active nodes in \( N(u) \). In the optimal 1-protection subset a node \( u \) must have at least one active node in its neighborhood. Thus the approximation factor is 33. Thus \( |A_1| \leq 33 \cdot |OPT_1| \).

4.3. Distributed Algorithm \( D \): 2 and 3-protection Subset Problems

In this section, we present a distributed algorithm for the \( p \)-protection subset problem where \( p \in \{2, 3\} \). The algorithm constructs protection subsets \( A_2 \) and \( A_3 \) for the 2 and 3-protection subset problems.

4.3.1. 2-protection.: We first run the distributed MIS [8] algorithm on \( G \) and let \( D_1 \) be the set of nodes returned by the algorithm. Nodes in \( D_1 \) are set as active. Each node \( w \in D_1 \) maintains a subset \( M_1(w) \subseteq N(u) \) which is initialized to null. Node \( u \) adds at most five nodes from its neighbors to \( M_1(u) \) to give the required protection to the nodes in \( N(u) \setminus M_1(u) \). In the following we elaborate on how neighbors are inserted in \( M_1(u) \).

Beginning with \( M_1(u) = \emptyset \), at first, \( u \) selects an arbitrary neighbor node \( v \in N(u) \) and adds it to \( M_1(u) \).

With \( M_1(u) = \{v\} \), node \( u \) checks whether all nodes in \( N[u] \setminus M_1(u) \) are covered by \( M_1(u) \). Otherwise, \( u \) selects the closest (in terms of Euclidean distance) node \( v_r \in N(u) \) such that \( v_r \notin E \) (ties are broken arbitrarily). \( v_r \) is added to \( M_1(u) \) (i.e., \( M_1(u) = M_1(u) \cup \{v_r\} \)).

Now \( u \) checks whether \( M_1(u) \) covers the nodes in \( N[u] \setminus M_1(u) \), if not, then \( u \) continues adding nodes \( v_r \in N(u) \) to \( M_1(u) \) such that \( v_m \notin E \), \( v_m \in M_1(u) \) and \( v_r \) is the closest (in terms of Euclidean distance) to some node \( v_m \in M_1(u) \). \( v_r \) is added to \( M_1(u) \) and \( u \) checks whether \( M_1(u) \) covers \( N[u] \setminus M_1(u) \). If not, then the process is repeated.

Due to the property of unit disk graph we know that \( M_1(u) \) can contain at most 5 such nodes (independent nodes), in which case all nodes in \( N[u] \setminus M_1(u) \) are covered. Notice
that as \( u \) forms \( M_1(u) \), \( u \) is already protected by \( M_1(u) \). If \(|M_1(u)| = 1\) then \( u \) arbitrarily selects a node \( v \notin M_1(u) \) to ensure its required 2-protection.

All nodes in \( \bigcup_{u \in D_3} M_1(u) \) are set as active. Notice that as \( M_1(u) \)'s are formed following the above procedure, all nodes \( V \setminus (D_1 \cup (\bigcup_{u \in D_3} M_1(u))) \) are at least 2-protected. This is because a node \( v \in V \setminus (D_1 \cup (\bigcup_{u \in D_3} M_1(u))) \) belongs to \( N(v') \setminus M_1(v') \) for some \( v' \in D_1 \) and since \( v' \) computes \( M_1(v') \), \( v \) must be covered by \( v' \) and \( M_1(v') \). Hence \( v \) is 2-protected. Each node in \( \bigcup_{u \in D_1} M_1(u) \) is guaranteed to be at least 1-protected since these nodes are protected by \( D_1 \). In order to achieve 2-protection for these nodes, each node \( u' \in \bigcup_{u \in D_1} M_1(u) \) which is not 2-protected arbitrarily selects a neighbor to protect itself. These selected neighbors are set as active. Let \( A_2 \) denote the set of all active nodes in \( G \). Then the set \( A_2 \) protects the nodes in \( G \).

### 4.3.2. 3-protection

Given \( G \), we first run the distributed MIS [8] algorithm and obtain \( D_1 \) as before, and the nodes in \( D_1 \) are set active. Following the procedure of constructing \( M_1(u) \), each node \( u \in D_1 \) now constructs a subset \( M_2(u) \) of at most 10 nodes to provide 3-protection to all nodes in \( N[u] \setminus M_3(u) \). If \(|M_2(u)| = 2\) for any \( u \in D_1 \) then \( u \) arbitrarily selects a neighbor \( v \notin M_2(u) \) to provide its own 3-protection. However, any node \( u_i \in M_2(u) \) is at least 1-protected since \( u_i \) is protected by \( u \). In order to obtain 3-protection, each node \( u' \in \bigcup_{u \in D_1} M_2(u) \) which is not 3-protected arbitrarily selects at most two nodes to protect itself. Now all nodes in \( \bigcup_{u \in D_1} M_2(u) \) and their selected neighbors are set as active. Let \( A_3 \) denote the set of all active nodes in \( G \). Then the set \( A_3 \) provides 3-protection to the nodes in \( G \).

In the following, we give the distributed algorithm \( D \) for the \( p \)-protection algorithm that is executed at each node \( u \):

**Algorithm \( D \) for \( p \)-protection subset**

**Input:** An undirected graph \( G \) and an integer parameter \( p \in \{1, 2, 3\} \)

**Output:** Protection set \( A_i \), \( i \in \{1, 2, 3\} \)

1: Run Distributed MIS Algorithm [8]
2: Let \( D_1 \) be the MIS found
3: If \( u \in D_1 \)
4: If \( p = 1 \)
5: Select \( v \in N(u) \) to protect \( u \)
6: Set \( u \) and \( v \) active
7: Endif
8: If \( p \geq 2 \)
9: Construct \( M_{p-1}(u) \) of at most \( 5 \times (p-1) \) neighbors
10: If \( |M_{p-1}(u)| = p - 1 \)
11: Add \( v \in N(u) \), \( v \notin M_{p-1} \) to \( M_{p-1} \)
12: Endif
13: Set \( u \) and the nodes in \( M_{p-1} \) active
14: Endif
15: Endif
16: If \( u \notin D_1 \) and active
17: If \( u \) is not \( p \)-protected
18: Select \( p - 1 \) neighbors and set them active
19: Endif
20: Endif

//All active nodes form the \( p \)-protection subset \( A_p \)

### 5. Theoretical Properties of Algorithm \( D \)

In this section we provide the correctness of the algorithm, that is, we prove that it produces a \( p \)-protection subset for all the nodes in the underlying graph. We begin with the following observation without proof:

**Observation 5.1:** Any node \( u \in V \) must have degree at least \( p \) to achieve \( p \)-protection subset.

**Lemma 5.2:** Active nodes in \( G \) constitute the \( p \)-protection subset to the nodes in \( G \).

**Proof:** In Theorem 4.2, we prove the case when \( p = 1 \). For \( p \in \{2, 3\} \), we prove the Lemma by contradiction. Assume that there is a node \( m \in V \) that is not \( p \)-protected. Let \( p_1 = 2 \) and \( p_2 = 3 \).

We consider the following three cases:

**Case 1:** \( m \in D_1 \). Since \( m \in D_1 \) and not \( p_i \)-protected \((i = \{1, 2\})\) that means it could not form \( M_1(m) \) (for \( p_i \)-protection) of cardinality of at least \( p_i \) (also recall that its cardinality can be at most \( 5i \)) to obtain the desired \( p \)-protection. So \(|M_i(m)| < p_i \). On the one hand, if \(|M_i(m)| = 0\) then this implies \(|N(m)| = 0\), a contradiction because the degree of a node in connected graph \( G \) cannot be \( 0 \). On the other hand, \(|M_i(m)| = p_i - 1\) implies...
Case 2: \( m \in M_1(m') \) for some \( m' \in D_1 \). Node \( m \) is guaranteed to have 1-protection since it is connected to \( m' \in D_1 \). Then \( m \) selects at most \( p_i - 1 \) nodes from its neighbors if \( m \) is not \( p_i \)-protected, in order to have the desired \( p_i \)-protection. If it cannot find \( p_i - 1 \) neighbors to protect it, then \( m \) has degree less than \( p_i \), a contradiction.

Case 3: \( m \) is not set as active. Since \( D_1 \) is an MIS in \( G \), \( m \) must be connected to some node \( m' \in D_1 \). If \( m' \) could not form \( M_1(m') \) of size at least \( p_i \) to cover \( m \) then either \( m' \) has degree less than \( p_i \) or \( m \) will be included in \( M_1(m') \). In both cases, it yields a contradiction since the degree of \( m' \) must be greater than \( p_i \) in the first situation and in the second situation if \( m \in M_1(m') \) then \( m \) will be active. \( \square \)

In the following we prove the main results of the paper.

**Theorem 5.3:** \( |A_2| \leq 78.5 \cdot |OPT_2| \) where \( OPT_2 \) is a minimum 2-protection subset and \( A_2 \) is the 2-protection subset returned by our algorithm.

**Proof:** We prove our constant factor approximation by bounding the number of neighbors of a node \( u \) that can join in \( A_2 \) to provide 2-protection.

There are two different cases to be considered, \( u \in D_1 \) and \( u \notin D_1 \).

**Case 1:** \( u \in D_1 \). Thus there is no node \( v \in D_1 \) in \( u \)'s neighborhood. \( u \) selects a set of at most five nodes in \( M_1(u) \) to cover nodes in \( N(u) \setminus M_1(u) \).

From Theorem 4.1 \( u \) can have at most 23 nodes in its two-hop neighborhood which belong to \( D_1 \). Consider node \( v \in D_1 \) which is exactly two hops away from \( u \). Since \( v \) selects a set of \( M_1(v) \) from at most five independent nodes there can be at most \( |M_1(v) \cap N(u)| \leq 2 \) (see Figure 4 (a)). We show this by the standard area equation \[ A = \frac{1}{2} \pi R^2 \]. The area equation for the intersection (also known as "vesica piscis" or lens) of the two disks centered on \( u \) and \( v \) (where \( v \) is 2-hop neighbor of \( u \)) each with radius \( R \), is at most \( A = \frac{1}{2} (4\pi - 3\sqrt{3}) \pi R^2 \). Plugging \( R = 1 \) into the equation we get \( A \leq \frac{1}{6} (4\pi - 3\sqrt{3}) = 1.2283 \). Thus we can have at most \( 5/\pi \approx 1.2283 = 1.95 \) nodes. That is, \( |M_1(v) \cap N(u)| \leq 2 \). Node \( u \) selects at most 5 nodes from \( M_1(u) \) and each of the 23 nodes \( v \in D_1 \) can have at most 2 nodes selected from \( N(u) \) to form \( M_1(v) \). Thus most \( 23 \times 2 + 5 = 51 \) nodes from \( N(u) \) are used to provide 2-protection. However, each of these 51 nodes selects at most one node from its neighbor if it is not 2-protected. Thus we can have at most 102 nodes selected from \( N(u) \). Since the minimum 2-protection subset \( OPT_2 \) requires exactly 2 nodes selected from the neighborhood of any node in \( V \), \( u \) selects at most 51 times more nodes from \( N(u) \).

**Case 2:** \( u \notin D_1 \). The case is similar to the above except \( u \) can have \(|D_1 \cap N(u)| \leq 5 \) nodes in \( N(u) \), (see Figure 4(b)). Each of these 5 nodes \( v \in D_1 \cap N(u) \) can select at most 5 nodes from \( N(u) \) to form \( M_1(v) \). Then each node \( v' \in M_1(v) \) will be at least 1-protected. If it is not 3-protected, then it can select at most one node from \( N(u) \), totalling \( 5 \!+\! 5 \!+\! 5 \!+\! 5 \!+\! 5 \!+\! 5 \!+\! 5 \!+\! 5 \!+\! 5 \!+\! 5 \!+\! 5 = 55 \) nodes selected from \( N(u) \). Adding this to 102 nodes we get at most 157 neighbors selected from \( N(u) \). Since the minimum 2-protection subset \( OPT_2 \) requires exactly 2 nodes selected from the neighborhood of any node in \( V \), \( u \) selects at most 78.5 times more nodes from \( N(u) \).

Thus our algorithm achieves a constant approximation factor of 78.5 for the 2-protection subset problem. \( \square \)

|\(|N(u) \cap M_1(v)\)|| \leq 2
|\( |A_2| \leq 78.5 \cdot |OPT_2| \)

**Figure 4.** (a) If \( v \in D_1 \), \( v \in N_2[u] \cap N(u) \) then \( |N(u) \cap M_1(v)| \leq 2 \) nodes. (b) Node \( u \) can have \(|D_1 \cup N(u)| \leq 5 \) nodes.

Following the same proof technique given above, we provide the bound for the 3-protection case.

**Theorem 5.4:** \( |A_3| \leq 153.66 \cdot |OPT_3| \).

**Proof:** We have the two following cases:

**Case 1:** \( u \in D_1 \). Node \( u \) selects a set of at most ten nodes in \( M_2(u) \) to cover nodes in \( N[u] \setminus M_2(u) \). Thus \( |M_2(v) \cap N(u)| \leq 4 \) for some node \( v \in D_1 \) two hops away from \( u \). Therefore, following the same line of argument, we have at most \( 23 \times 4 + 10 + 2 \times (23 \times 4 + 10) = 306 \) nodes selected from \( N(u) \). Since the minimum 3-protection subset \( OPT_3 \) requires exactly 3 nodes selected from the neighborhood of any node in \( V \), \( u \) selects at most 102 times more nodes from \( N(u) \).

**Case 2:** \( u \notin D_1 \). Each of these 5 nodes \( v \in D_1 \cap N(u) \) can select at most 10 nodes from \( N(u) \) to form \( M_1(v) \). Then each node \( v' \in M_1(v) \) will be at least 1-protected. If it is not 3-protected, then it can select at most two nodes from \( N(u) \), totalling \( 5 \!+\! 5 \!+\! 10 + 2 \times 5 + 10 = 155 \) nodes selected from \( N(u) \). Adding this to 306 nodes we get at most 461 neighbors selected from \( N(u) \).
Since the minimum 3-protection subset $OPT_3$ requires exactly 3 nodes selected from the neighborhood of any node in $V$, $u$ selects at most 153.66 times more nodes from $N(u)$.

Thus our algorithm achieves a constant approximation factor of 153.66 for the 3-protection subset problem.

5.1. Time and Message Complexities

Exchanging indices and degrees with direct neighbors can be done in a constant time [15]. Line 1 in our algorithm $\mathcal{D}$ constructs a distributed MIS [8] in a unit disk graph which requires $O(\log^* n)$ time [8] where $n$ is the number of nodes in $G$. The algorithm in [8] can be applied in any growth bounded graph, where growth boundedness means that there is a constant number of independent nodes in some fixed hop neighborhood. Line 2-20 of algorithm $\mathcal{D}$ can be executed in constant time since nodes only perform local computation which is free [15]. Thus algorithm $\mathcal{D}$ needs $O(\log^* n)$ time.

Each node broadcasts a single message containing its index and degree to its direct neighbors. As mentioned in [8], the algorithm for generating an MIS requires at most $O(|E| \log^* n)$ messages. Also nodes in $D_1$ inform their neighbors whether they will be active which totals at most $O(n)$ messages. Thus the message complexity of algorithm $\mathcal{D}$ is $O(|E| \log^* n)$.

This suggests the following theorem:

Theorem 5.5: For unit disk graphs the $p$-protection subset problem ($p \in \{1, 2, 3\}$) can be solved within a constant factor in $O(\log^* n)$ time and with $O(|E| + \log^* n)$ messages.

6. Conclusion and Open Problems

Providing protection to sensors by their neighbors in a sensor network is an interesting problem. The need to protect sensors arises because they are vulnerable to attacks, failure or damage while they monitor objects or provide coverage to a certain area. By having some protector sensors around others allows us to know the status of these sensors and take appropriate actions (replacing the dying and faulty sensors, if possible) in case those sensing sensors fail or run out energy.

In this paper, we have studied the problem of protecting the sensors by one another namely, the $p$-protection subset problem, and provided a distributed solution with a constant approximation factor. To the best of our knowledge, this is the first constant factor approximation ratio to $p$-self-protection subset problem which is significantly better than the previous $O(\log n)$ approximation ratio.

We have identified some open problems that result from this work. First, it will be interesting to obtain a stronger and smaller constant factor for the $p$-protection subset problem. We do not know whether it is easy to have a ‘small’ constant factor bound for the $p$-protection subset problem when $p > 3$. It is easy to see that following our procedure we can obtain solutions when $p > 3$, however, the constant factor will be much larger as the values of $p$ increase. Providing a ‘small’ constant for arbitrary values of $p$ is a challenge.

Our constant factor approximation is large because we focus only on the worst case behavior, however we believe that the approximation ratio will be much smaller in general cases. Besides, the $p$-protection subset we find is not necessarily connected, thus, finding a connected $p$-protection set, that is, the nodes in the protection set induce a connected subgraph, would be interesting. Another challenging open problem is to find a disjoint family of $p$-protection sets such that each of the sets can be used for a round and then can be replaced by another set for the next the round, and so on, in order to properly balance energy consumption.

References


Ehsan Valavi*, Hamid Reza Asaadi* and Ahmad Sabouri*

* School of Electrical and Computer Engineering,
University of Tehran, Tehran, Iran
Email: {e.valavi, hr.asadi, a.sabouri}@ece.ut.ac.ir

Abstract—The raising development of communication theory and computer science has opened new challenges in the field of sensor networks. Since we have many constraints on the size of sensors and power consumption issues, one of the most important problems is efficient resource allocation and sharing without a central coordinator as well as limited sensing or processing. In this paper, we are going to address the problem of resource allocation in WSN to a simple machine learning problem called Multi-Armed Bandit. Actually, we modify the standard form of the problem to a dynamic condition in which the resources are available in a covariate. Then, we propose a simple and possible model of this algorithm which it seems to be suitable for embedding on sensors.

I. INTRODUCTION

The raising development of communication theory and computer science has opened new challenges in the field of sensor networks and initiated a great rush of attention to Wireless Sensor Networks (WSN) and some other related problems. WSN consist of a large number of spatially distributed sensor nodes with heterogeneous sensing and computational capabilities which communicate over wireless channels for collaborative information processing. WSN have variety of potential applications such as maintenance and control of complex systems, military surveillance, environmental and habitat monitoring and etc.

Since we have many constraints on size of sensors and power consumption issues, one of the most important problems of sensor networks is efficient resource allocation and sharing without a central coordinator as well as limited sensing or processing.

The classic problem of resource allocation (i.e. scarce resources to minimize cost and maximize capability of trying on every invaluable aspect of using sensor nodes) has been a well-studied one. The broad field of operational research can be described as the study of optimal resource allocation. Scheduling, assignment, bin-packing, timetabling and etc. are some typical resource allocation problems which have solutions from various fields of dynamic programming, combinatorial optimization, game theory and etc.

These problems lead us to use optimized methods in design of sensor nodes. Thus, various methods have been proposed to solve these problems. Many of these proposed methods are based on intelligent selfish nodes learned by reinforcement learning techniques to use the resources in a certain way. In [1] a Self-Organized Resource Allocation (SORA) approach is presented to achieve these aims. In this approach, each node adapts its operations in response to the feedback from some properties of bargaining payments using reinforcement learning.

Lim et al. [2] proposed the effectiveness of an Adaptive Distributed Resource Allocation (ADRA) scheme. This scalable scheme specifies simple local actions to be performed by individual sensor nodes. It can help to have more control over the neighbors’ behaviors.

In market-based techniques [1] [3] [4], resources are allocated through trading activities between agents. A seller tries to maximize its earnings whereas a buyer seeks to minimize its spending. Certain strategies are used to control habits of agent through the propagation of price information. Another interesting approach to the resource allocation problem is the auction and bidding techniques for allocating resources to tasks, such as combinatorial auction [5] [6] and coalition formation [7].

In this paper we address the problem of resource allocation in WSN to a simple machine learning problem called Multi-Armed Bandit [8]. This problem is based on an analogy with a traditional slot machine with more than one lever. Each lever makes a reward drawn from a distribution associated to that specific lever. This specific distribution is defined as a stochastic process which can be used as a model for the resource availability.

Ref. [9] discusses some application of multi-armed bandit in designing finite dimension algorithms for Hidden Markov Model type of this problem. It seems to be useful in some other application in which nodes have more than one task.


The multiarmed bandit formalism has been used to study a number of diverse applications including adaptive routing, clinical trials, real-world data retrieval problems with redundant sources and many aspects of optimization [10].

In this paper, we apply this algorithm to WSN resource allocation problem. The optimality of the network in this paper is considered as the rate of packet transmission for each node. In the worst case, the network throughput will be zero and no
node can perform a transmission.

The rest of this paper is organized as follows. In section 2, we will define the multi-armed bandit algorithm and its requirements. In addition, we propose a simple and possible model of this algorithm which can be applied on sensors in a WSN. Then, we define some requirements of our simulation case. Section 3 will present the simulation results coming from our experiments. We use some simplifying conditions to apply the algorithm on the sensors in the simulated environment. Finally, section 4 covers the conclusion and an overview of the tasks which would be done in the future.

II. SYSTEM MODEL

In this section, we identify the multi-armed bandit problem statement and its application in WSN resource allocations. We also define criteria of success in resource allocation (using action reward calculation).

It should be taken into consideration that nodes consist of many parts including sensing and transmitting equipments, battery and some other peripherals. Since each node has at least a radio transmitter and a packet queue, it is obvious that optimal routing in free channels would make the transmission process faster. Actually, the optimality is to minimize the number of queued packets and this can be achieved through utilizing maximum number of free channels in order to send data.

A. Multi-Armed Bandit Problem

The tradeoff between exploration and exploitation [12] is often studied in more general and standard models such as Markov Decision Processes. The multi-armed bandit is an interesting machine-learning problem based on an analogy with a traditional slot machine but with more than one lever. At each slot, users pull one of the levers and get their reward coming from a prior known distribution for that lever. In this problem, the main issue is how to find a balance between discovering new knowledge (exploration) and maximizing reward based on existing knowledge (exploitation). Therefore, the aim of a gambler is to maximize the sum of collected rewards through iterative pulls. It is classically assumed that the gambler has no initial knowledge about the levers.

We can see the lever selection as an action done by gambler and it means that the gambler faces with a choice between some different actions. This set of actions can be represented as $A = \{a_1, a_2, ..., a_n\}$ where $n$ is the number of actions. After each action selection, the gambler plays and receives a reward derived from the probability distribution associated with the selected action.

So we face an optimization problem and we have to minimize the following equation ($d(t, T)$ which represents the difference between the observed and the maximum available rewards) by our algorithm:

$$d(t, T) = T \times \mu_{max} - \sum_{i=1}^{T} \rho_t$$

in which $T$ is the number of actions done by gambler, $\rho_t$ is the reward coming from these selections over the time and $\mu_i (i = 1, 2, ..., T)$ is the maximum reward of each action done by the gambler.

These assumptions make it possible to model our problem of resource allocation with the general model of multi-armed bandit by assuming that the resources are rewards coming from the actions done by the nodes (i.e. pulling lever or playing any action).

B. Algorithms

We consider only the $\epsilon$ greedy strategy (first described by Watkins [13]) in this work. This action selection algorithm is one of the simplest and the most widely used strategy to solve the bandit problem. This algorithm selects the action with the highest expected reward based on the current parameter estimated with probability $(1 - \epsilon)$. Therefore, a random action is selected with probability $\epsilon$. These types of random action selections help us to mutate and escape from any local optimum results. In a static environment, $\epsilon$ value determines the balance between exploitation and exploration.

Since all rewards in a dynamic environment change in all iterations, selecting the greedy action based on current estimated values can be seen as exploring the change of this action.

C. Sensor Modeling

We define a new extension of this simple method with covariate resources. The fact of changing the best action over time leads this model to lack any optimal policy [11]. In this system, nodes face a covariate coming from the measurements at each round. In this paper, the covariate is assumed as $x(t) \sim N(\mu, \sigma^2)$, and the reward is modeled by a linear function of the covariates passing through the system with impulse response $h(t)$. Using $h(t)$ is one of the most important points which leads to a better design of systems based on the system requirements and expected node behaviors. We can define $h(t)$ to decrease errors in our measurements. The following equations represent the mechanism for reward calculation in an analytical form:

$$r_{\alpha_i} = \beta_0 + \beta_1 \times (x(t) \times h(t)) + \epsilon_i, \quad \epsilon_i \sim N(\mu, \sigma^2),$$

$$x(t) \times h(t) = \int_{\tau=-\infty}^{+\infty} x(t-\tau) \cdot h(\tau) d\tau$$

In the case of rewarding the agents with $x(t)$, we have $h(t) = \delta(t)$ in which $\delta(t)$ is the Dirac delta function. Nevertheless, in some type of resources allocation problems like capacity evaluation of communication channels we face some errors in the measurement. Thus, the measured $x(t)$ may contain some errors in measurement caused by channel dynamic behavior (e.g. varying channel capacity over time) which can be presented as

$$\mu_{max} = \max \{\mu_1, \mu_2, ..., \mu_T\}$$

where

$$\sum_{t=1}^{T} \rho_t$$

is the number of actions.
\[ x(t) = \hat{x}(t) + n(t) \]  

(5)

where \( n(t) \) is a noise (i.e. the measurement errors) function. From the signal analysis perspective, we can use some analytical transforms in order to investigate the noise power in frequency domain. This can help us reduce the power of noise which causes more reliability on the measured parameters [14]. The power of this noise can be reduced by reshaping the \( x(t) \) with suitable selection of \( h(t) \).

### III. EXPERIMENTS AND RESULTS

This section discusses about bandwidth sharing and allocation as a simple problem of allocating resources in WSN. We also investigate the results of our simulations. A simple filter is used in order to simulate the Multi-Armed Bandit problem. Figure 1 depicts the general schema of the process.

Assume a surface which the nodes are distributed on as the simulation environment. Therefore, in the first step, we release the nodes in the surface using a uniform probability distribution to avoid any prior known map. The average distance between the nodes, the coverage range of them, the number of neighbors and the number of available channels are our variables associated with the nodes in this simulation. The distances between the nodes and the average number of the neighbors have a relation and also the same behavior. So we show the effect resulted by one of them.

In the beginning of simulation, we consider an initial offset to each node which let us partition the nodes into some groups with the same size. This grouping allows us to have more control over the node behaviors.

Another important task in the simulation is to define a time between two successive sending in order to avoid any mistake which lead our nodes to be identified as malicious nodes in the network. So the nodes send their data packets at the end of the time interval according to their offsets. Actually it is the time between two successive sending.

The tests show that if the nodes in the networks send their data simultaneously and we do not consider enough channels for the nodes, the network performance goes down dramatically. This condition is not a common event in the real world and we can assume that they send their data separately based on a probability distribution.

In the sending process, each node selects one of the channels. This task is done by the two algorithms, Multi-Armed Bandit selection method and random channel selection.

If the channel selection mechanism is the random selection then the sending process begins and the nodes perform their tasks in \( N \) different cycles. When a process ends, the next process starts and so on. If the selected channel is occupied by the others, the node must wait for the next cycle to select a channel and send its packet.

In the modified version of Multi-Armed Bandit problem used in this paper and discussed earlier, the nodes give a reward to the correct selected channels in their memory. This reward is calculated form an integral which correspond the systems property. The integral is calculated over an interval with a prior known range. It can help us to measure the suitability of resources more accurately. In fact we try to reduce the power of measurement errors by use of integral in order to push the power of these errors to its average (it is more suitable when the average of noise magnitude is zero). The system equation is shown below:

\[
h(t) = u(t) - u(t - m). \tag{6}
\]

\[
u(t) = \begin{cases} 
0 & t < 0 \\
1 & t \geq 0 
\end{cases} \tag{7}
\]

In the above equation \( m \) is the length of intervals. The interval is defined as the time distance between data cycles and cycle is the minimum time between two transmissions in the network or by individual nodes.

In our experiences we use almost two thousands nodes in an area about ninety thousand meters. The variable factors are node coverage range and the distribution of them. We use uniform and Gaussian distributions for our experiment.

Number of channels to use by the sensors is varying based on the average neighborhood of the nodes. We try some of them in a certain way.

The number of unsuccessful packet transmitting of a node is available in figure 2. The coverage range is about ten meters and the average time for packet sending is two cycles. Also, the average number of neighbors is about seven nodes.

In this simulation we run the code over time. Also, in the other experiments we change the number of channels over
time. In every experiment, we simulate the map in hundred cycles. The result is available in the figure 3.

Finally, we can observe the effect of increase in average count of a node on the service quality. As we expect, the number of unsuccessful packet transmissions will grow with increase in average neighbors’ count of nodes (figure 4). Since we do not guarantee the success of the packet transmissions, we have no observations on the status of the receivers in this work. Thus the number of unsuccessful packet transmissions will be excessively low in the case of small neighbor count.

IV. CONCLUSION

In this paper we address the problem of resource allocation in wireless sensor networks to a modified version of Multi-Armed Bandit problem. This modified version is also available to be more complex in order to use in many more complex devices. In our experiments we found out that our proposed algorithm takes many advantages even in its simple form. The result shows that at least the performance can be doubled by our algorithm versus the random channel selection. This method can be modified to use in other resource allocation problem by designing appropriate filter. As a future work we can use the reliability or the usefulness of data as a reward mechanism.

REFERENCES


DCOSS Workshop:

Second International Workshop on Sensor Network Engineering (IWSNE)
Even though this is officially the Second International Workshop on Sensor Network Engineering, it is actually the third in a row. After an initial event in San Francisco during DCOSS 2006, which was mainly restricted to European and American researchers, the first official instance took place on the Greek island of Santorini, hosted by DCOSS 2008. And we have a new feature this year: while all of last year’s papers were invited, we had an open paper submission process this time. After a thorough peer-review process, finally six papers were selected to form this half-day workshop on several aspects on engineering methods and tools for designing and implementing wireless sensor networks.

While SOAs are an attractive tool for enterprise application development, they also typically require a lot of resources. In their paper "On the Integration of Service-Oriented Architectures and Sensor Networks", Nils Glombitzka and his co-authors present an approach for using SOA technologies for programming complex sensor network applications. Vlad Trifa et al. work on similar problems and also search for solutions for the simple programming of heterogeneous sensor networks. In their contribution "Design and Implementation of a Gateway for Web-based Interaction and Management of Embedded Devices" they propose to equip all networked devices with a RESTful interface and thus enabling access through URIs.

The paper "ESAC: Supporting the Configuration of Simulation Models and Applications for Wireless Sensor Networks" by Jochen Koberstein et al. tackles the problem that simulation results and real-world behavior of sensor networks often differ a lot. The approach uses evolutionary algorithms to develop simulation model configurations which come as close as possible to real-world situations. Paul Grace and his co-authors address the problem of the often dynamic operational environment of sensor networks. Today’s middleware approaches are often not flexible enough to react to sudden changes; in their paper "Middleware Support for Dynamic Reconfiguration in Sensor Networks" they present a reflective middleware solution for coordinated dynamic re-configuration of middleware behavior across nodes in a sensor network.

Ioannis Chatzigiannakis et al. work on making sensor network data globally available. In "A Peer-to-Peer Framework for Globally-Available Sensor Networks and its Application in Building Management", they present the architecture and implementation of WebDust, a software platform for managing multiple, heterogeneous and geographically disparate sensor networks.

Finally, "TinyAID: Automated Instrumentation and Evaluation Support for TinyOS" addresses the need for debugging and analyzing the operation and results of sensor networks. Christoph Weyer and his co-authors have developed TinyAID, a tool that supports automated instrumentation and evaluation of TinyOS-based distributed applications.

We thank the program committee for putting together this interesting program and hope that the presentations find a huge audience during DCOSS 2009 in Marina del Rey!
On the Integration of Service-Oriented Architectures and Sensor Networks

Nils Glombitza, Dennis Pfisterer, and Stefan Fischer
Institute of Telematics, University of Lübeck
Ratzeburger Allee 160, 23538 Lübeck, Germany
Email: {glombitza, pfisterer, fischer}@itm.uni-luebeck.de

Abstract—Currently, two major trends are gaining momentum: Service-Oriented Architectures (SOA) based on Internet standards are widely deployed in industry and Wireless Sensor Networks (WSNs) are becoming an integral part of the Future Internet. Combined, these two trends lay the groundwork for a new class of applications where all kinds of devices ranging from simple sensor nodes to large-scale application servers interact to drive business processes not possible before. That way, data stemming from a WSN monitoring assets in a port area influence the control flow of a business process in real-time or even trigger a business process. To achieve this level of integration, WSNs must seamlessly interoperate with existing widely deployed SOA technologies such as XML, Web Services, and the Business Process Execution Language (BPEL) to name only a few. However, due to their high resource demand, these technologies are hardly applicable in resource-constrained WSN environments. In this paper, we present a survey of existing approaches and introduce our technique to seamlessly integrate WSNs into SOA environment using only very few resources.

I. INTRODUCTION

Embedded systems are constant companions of our daily life. Augmented with sensors and a wireless communication interface and optimized for low-power, low-cost and a small form factor, a new class of resource-constraint devices, so-called sensor nodes, has emerged. A – potentially large – number of sensor nodes form Wireless Sensor Networks (WSNs) which are beneficial in situations where human observation is hardly possible or where wired sensing systems are difficult or expensive to use. Possible application scenarios are habitat monitoring and health-care as well as applications in telematics, security and military domains [1]. Due to their special nature, WSNs are in general highly specialized networks built and operated by domain experts for a dedicated application purpose.

Now that WSNs are gradually adopted in industry applications, it is compulsory that they blend well with current and future IT infrastructures without the need for specialized personnel for their installation, management, and operation. Instead, they should be easily deployable and usable by any business process that requires the interaction with a WSN. Services offered by single sensor nodes or the sensor network as a whole must therefore be accessible using well understood and widely accepted techniques. However, as adumbrated above, the current state of the art in WSN research is to design application-specific solutions and only few approaches exist that tackle the interconnection of WSNs and off-the-shelf IT infrastructure.

In more general terms, the integration of legacy systems into a coherent IT infrastructure is a major issue in contemporary business environments. To overcome these issues, the concept of Service-Oriented Architectures (SOA), which has evolved in the past few years, is a supreme solution. In a SOA, applications no longer consist of monolithic and closely coupled software systems. Instead, they are flexibly composed of existing services on demand, e.g., to compose complex business processes instantaneously to adapt to customer needs [2]. This flexible architectural concept allows a natural and seamless integration of WSNs where applications running in a sensor networks provision data, features and actions as services in a SOA environment.

In this paper, we consider the predominant implementation of a SOA, which is based on open Internet standards such as XML [3], [4], Web Services [5], and the Business Process Execution Language (BPEL, [6]) to name only a few. These technologies inherently target server-class computers where processing of XML documents, running application servers, and handling complex business logic is easily accomplished. However, they overburden the resources of sensor nodes, where memory, processing power, energy, and bandwidth are severely limited, by several orders of magnitude.

Accordingly, the research question considered in this paper is how WSNs can fully partake in such a SOA environment based on Internet standards despite their limited resources.

In the following, we first present individual research challenges (cf. Section II) and then provide an in-depth survey and taxonomy of existing approaches in Section III. In Section IV, we argue that the current state of the art is not sufficient and introduce our approach to integrate WSNs into a SOA environment using only very few resources. Finally, Section V concludes this paper with a summary and an outlook for future work.

II. RESEARCH CHALLENGES

The use of SOAs is driven by the very real demand of companies and businesses to get timely access to their data and the need to adapt the flow of information to new business processes quickly. This is the result of a process where data of companies are becoming equally important assets as the goods and services offered by them. The demand to integrate
real-world data into business processes can be observed by the overwhelming success of the RFID technology. In contrast to RFID, WSNs can actively interact with business systems and can hence support innovative applications not possible before. The integration of WSNs is therefore a logical next step in Enterprise IT.

Consider the application example of a port (cf. Figure 1) where containers enter the port by train, stay for a certain period of time on the port’s area and are then loaded on cargo ships and vice versa. In this scenario, the goods in the containers are equipped with sensor nodes.

Once the train is assembled, the sensor nodes transfer their data via a WSN gateway on the locomotive to the Enterprise IT system of the railroad company. From here, this information is transferred to the port’s system. By interaction with the sensor nodes, the IT system has a real-time view of the occupied and free positions and can determine suitable ones for the containers that are about to arrive. It can even take security precautions, e.g. ensuring that explosive goods are not near to each other. Following, information about containers and goods, which are about to be loaded onto a ship, is transmitted to the shipping company. The ship’s IT system can hence check if the correct containers are loaded by querying the WSN. This is just one example of how the integration of WSNs into a business process may enhance the quality of real-world and IT processes.

To achieve this level of integration, individual sensor nodes must be able to receive and send messages that are understood by Enterprise IT servers running SOA technologies. As SOAs are mostly implemented using Internet standards, such as Web Services, these are the technologies that must be supported in WSN environments. Figure 2 shows the full Web Service Technology Stack and mentions the most important standards used nowadays. This includes well-known technologies and protocols such as TCP/IP, HTTP, SMTP, XML, and SOAP. A typical Web Service implementation may use TCP/IP for routing and reliable transport, HTTP on the application layer to correlate request- and response-messages and SOAP to pass on the XML-encoded messages via HTTP.

While these technologies are virtually omnipresent in the Internet, they are typically not available in the resource-constraint environment of WSNs where energy, bandwidth,
computational power, RAM, and ROM are scarce resources. This requires energy-efficient messaging where the overall duration of transmitting and receiving is kept to a minimum. Hence, transmitting bulky SOAP/XML documents is not an option. Instead, compression techniques or optimized encodings are mandatory. Furthermore, such documents may easily overburden the RAM of sensor nodes by several orders of magnitude. No in-memory processing of XML should therefore be performed also due to the limited computational resources and amount of code-ROM where XML-parsers or other toolkits do not fit into. Similarly, protocols such as HTTP, SMTP, or TCP are typically avoided in WSNs due to their resource-consumption.

As a result, to use standard-compliant Web Services and SOAs based on Web Services in WSNs, the following techniques are required:

1) a compact encoding of SOAP/XML messages and a
2) lean transport binding for WSNs.

In the following, we provide an overview and a taxonomy of existing approaches on all layers of the Web Service technology stack in the context of Enterprise IT and WSN environments in Section III. We discuss pros and cons of the individual techniques and show that there is still a gap that hinders the productive and transparent use of Web Services in mixed Enterprise IT and WSN systems. In Section IV, we present our approach of how WSNs can fully partake in such a SOA environment based on Internet standards despite their limited resources.

III. TAXONOMY OF EXISTING APPROACHES

In this section, we introduce related work from the Web Service, Enterprise IT and WSN research communities. We have clustered related work according to the individual layers of the Web Service technology stack as shown in Figure 2. This survey includes each layer except the Quality of Service layer which is postponed to future work. In the following, Section III-A introduces the Business Processes layer, Section III-C is about the Description of Web Services (messages and operations), Section III-B deals with the discovery of Web Services, Section III-D discusses the (de-)serialization of data, and Section III-E concludes with aspects on the transport and delivery of Web Service messages.

A. Business Processes

Due to the high level of economical globalization and constantly changing conditions on the markets, enterprises must change or adapt their business processes permanently. Of course, this includes the adaptation of IT processes and applications. Since the 1980s, business process or workflow management systems address exactly this problem.

Unlike classical programming languages like C/C++, business process modeling languages compose existing services to business processes in a declarative manner. This enables a fast and flexible process redesign without changing source code. Furthermore, a lot of these languages can be used directly by business personnel thus saving time and costs of the modeling process.

In the context of a service oriented architecture, the business process modeling languages BPEL (Business Process Execution Language) [6] and ebXML/BPSS (Business Process Specification Schema of “Electronic Business using eXtensible Markup Language”, [7]) have gained widespread adoption. Both languages describe and execute business processes as technical implementations on top of the underlying Web Service technology. Other than in the Enterprise IT environment where ebXML and especially BPEL is very successful, there is hardly an approach trying to make use of the business process modeling paradigm in sensor networks.

Spiess et al. [8] use BPEL with sensor networks to design an application for monitoring hazardous goods. Unfortunately, this work does not describe which part of the application is exactly designed with BPEL and if it is used in the sensor network itself or only on the backend computers.

The GWELS (Graphical Workflow Execution Language for Sensor Networks, [9]) language and toolkit is a first step to integrate sensor network applications or services offered by sensor nodes into business processes. Even though GWELS processes can be integrated into a Web Service-based SOA over a Web Service connector, the communication between the GWELS processes and the services provided by the sensor nodes is based on proprietary communication technologies.

Amundson et al. [10] present an approach enabling the graphical design of sensor network applications. Like GWELS, this approach is also based on a proprietary communication middleware and applications are realized as graphs of services. The execution is triggered by the detection of a physical phenomenon recognized by a sensor. As further steps, the authors mention the use of XML-based Web Service standards, which are adapted to the limited hardware capabilities of sensor networks, as future work.

B. Discovery

Service discovery protocols play a key role in SOAs as well as in sensor networks. While a lot of solutions for IP-based networks and Web Services are already developed, there is still demand for developing solutions to discover services in sensor networks.

In SOAs based on Web Services, UDDI (Universal Description, Discovery and Integration, [11]) was designed to play a key role for service discovery in the Internet. UDDI is a centralized repository where every service provider has to register the provided services. Thus, there is a large overhead in terms of the administration time and effort and, as a result, UDDI did not gain a high popularity in the Internet. Furthermore, such a centralized approach is not applicable for unmanaged and highly dynamic sensor networks.

Web Service Dynamic Discovery (WS-Discovery, [12]) is a decentralized service discovery approach that transmits SOAP messages usually using UDP multicast packets. When entering a network, service providers broadcast a hello-message while service providers leaving a network broadcast a bye-message.
In addition to this provider-driven discovery, service clients can discover service providers by sending discovery requests including a service type to a service group addressed by a multicast IP address. Service providers which offer this type of service answer directly to the client. As a decentralized and self-managed approach, WS-Discovery is applicable for the use in sensor networks. But to transfer this approach to the resource-constrained WSN environment, two challenges need to be addressed. UDP and even the underlying IP protocol are too resource-demanding for a lot of sensor network platforms. But the WS-Discovery standard does not limit the selection of the transport protocol to UDP and every multicast protocol, which is capable of transferring SOAP messages, can be used. The second challenge is the size of the exchanged SOAP messages that exceed the resources of sensor nodes. Both problems are addressed by our approach presented in Section IV.

Another approach to discover services in ad-hoc or un-managed networks is UPnP (Universal Plug and Play). The operating mode of UPnP is very similar to WS-Discovery, but it is not based on Web Services. Dobrescu et al. [13] describe how UPnP can be used for service discovery in sensor networks. But UPnP requires IP which is not the preferred protocol for every WSN application and furthermore, it is not applicable on every resource-constrained hardware platform. For the same reason, the also IP-based Service Location Protocol (SLP) which is used in some sensor networks [14] does not represent an all-encompassing solution.

C. Description

To enable application development based on the composition of services of different communication partners, a platform-independent description of the communication interface of services is required. Such a description has to contain all information, which is required to use a remote service without additional knowledge about its implementation. Furthermore, service descriptions should be automatically readable and interpretable by computers.

The Web Services Description Language (WSDL, [15]) is an XML-based language that enables the abstract description of service interfaces including its operations and technical information about the communication endpoint of that service. The abstract part of a WSDL description contains the specification of interfaces, operations, and data types while the concrete part describes the binding and the service. The interface describes the elementary functionality of a Web Service and embraces a set of operations defined by XML messages. The exchanged messages respectively the data types are usually specified using XML Schema [16], which is a XML language defined by the W3C for defining the grammar of XML documents.

In the concrete part of WSDL, the binding defines which transport protocol (transport binding) is used for exchanging the Web Service messages (see Section III-E). Furthermore, the message protocol of the Web Service messages is specified. Usually, SOAP [17] (cf. Section III-E) is used as message protocol, but arbitrary others can be used, too. Finally, one has to specify where a Web Service instance is physically available. This is done by using the service element embracing a set of service endpoints.

Even though there are some Web Service frameworks available in the Enterprise IT environment that enable the dynamic integration of different Web Services with different WSDL descriptions at runtime, usually the relation between service consumer and provider is fixed at runtime. Here, “fixed” refers to the abstract part of a WSDL description and the binding but not to the endpoint addresses, which usually changes dynamically at runtime. Hence, there are no open researches issues considering the transfer of this technology to sensor networks. Nowadays, when changing a sensor network application, a reprogramming of the sensor nodes is required, anyway.

D. Messaging

A key element of WSNs and SOAs alike is to solve a problem as a coherent swarm of devices and cooperating services. Since the devices and services do not share a common memory, communication between is the enabling means to solve the global task. As a result, data held in memory, must be serialized into payload of network packets and vice versa. This is required to provide processing of application data in conventional programming language constructs (e.g., structs in the C programming language or classes in Java). In web-service-based SOAs, the ubiquitous means for exchanging information is to encode in-memory data structures as XML documents.

However, the devices involved in our scenarios are of highly different natures ranging from resource-constrained sensor nodes to resource-rich server-class systems. Since XML documents are very bulky, even small ones overburden the available bandwidth and memory offered by typical sensor nodes by orders of magnitude. In addition, given that receiving and transmitting data are the most energy-hungry operations in a WSN [18], [19], the overall broadcasting duration must be minimized to save energy. To achieve this, basically two options exist: reducing the number of transmitted messages and minimizing the length of individual transmissions. While the former is highly specific to each application, the latter can be generally applied. Consequently, the representation of application data as payload must be optimized to require only a minimum of bits.

To implement the conversion from in-memory data structures to payload, two fundamentally different approaches exist. In the first approach, developers handcraft data structures and manually implement the mapping from and to payload. The second approach is to describe the application’s data types in a platform-independent manner and to transform this specification into platform-dependent code for multiple target platforms and languages. In the following, we discuss variants of both approaches.

1) Approaches Based on Handcrafted Code: In the context of WSNs, the application’s data types are in the majority of
cases implemented as nested data structures directly in the target programming language [20]. An approach frequently used in WSN software is that the payload is an exact, bitwise copy of the in-memory representation on the sensor node. Although widely used due to its simplicity, this approach has various, severe drawbacks: it relies on the assumption that different compilers for different devices represent data structures in exactly the same manner so that data serialized from memory on one device can be de-serialized on another. This assumption often holds inside the sensor network where exactly the same program is running on identical hardware. However, due to the increasing standardization of the radio interface, even inside the WSN, device heterogeneity is a non-negligible issue.

The network types [20] technique addresses this problem by proposing a programming language extension to nesC [21] that introduces new keywords to the language. Instead of using the traditional C-language constructs struct and union, developers use nx_struct and nx_union and a number of new data types (nx_int8_t, nx_uint8_t, nx_int16_t, etc.). A modified version of the nesC compiler translates these new language features into standard C code before it is actually compiled. This approach specifically enables the cooperation of sensor nodes that are programmed using nesC by mitigating the effects of different memory alignments and endianness. Yet, the designed data structures are still confined to the nesC programming language. Hence, this approach is neither suited for heterogeneous WSNs nor for the integration with traditional networks. Custom data types that are created using these new keywords are subject to a number of restrictions. The most important one is that nx_struct and nx_union may not contain bit-fields. Bit-fields are commonly used to decrease amount of memory required by a data structure. When this in-memory representation is directly copied into the payload of network messages, this also reduces the payload length. As a result, this common optimization cannot be used.

The External Data Representation (XDR, [22]) standard specifies an architecture independent data encoding to ease the data exchange between heterogeneous computers. XDR is typically available as a library that is used by programs to convert data from and to a common encoding. It defines a number of XDR data types and their mapping to this common encoding. XDR is typically used by Sun’s Remote Procedure Call (RPC, [23]) framework, which is used by the well-known Network File System (NFS, [24]). Despite its usefulness for desktop and server class computers, XDR is not suited for resource-constrained devices and the resulting encoding is not optimized for size.

A textual specification of the payload format is another alternative. Hereby, the meaning of each byte is described in a human-readable form. Common examples for this procedure are Internet protocols, which are defined in so-called Requests For Comments (RFCs). Figure 3 depicts an excerpt from RFC 768 [25] that standardizes the User Datagram Protocol (UDP). This is a typical example of how protocol headers (that are the payload of other protocols such as IP) are defined. Developers manually implement code that copies the content of in-memory data structures to the correct positions in the payload as defined in the specification. In doing so, they must take care of different memory alignments and endianness manually.

![Fig. 3. User Datagram Header Format](image)

2) Code Generation and XML Compression Techniques: The second approach is to describe the application’s data types in a platform-independent manner and to transform this specification into platform-dependent code for multiple target platforms and languages automatically. The generated code comprises the language- and platform-specific data types and routines that (de-)serialize them to and from a common encoding. A number of techniques exist that are widely used in various disciplines of computer science.

Walther et al. [26] present an approach that automates the manual implementation of textual protocol header definitions as shown in Figure 3. They propose a technique to generate data types and (de-)serialization code automatically from protocol headers that are defined using this ASCII-art like syntax. However, the underlying input format used by this approach only provides a limited expressiveness for the definition of complex, nested data structures compared to languages such as XML Schema or ASN.1.

The Abstract Syntax Notation One (ASN.1, [27]) provides a language for describing data structures in a manner very similar to XML Schema. Besides the ASN.1 language, encoding and decoding schemes have been standardized, among them the XML Encoding Rules (XER, [28]), the Basic Encoding Rules (BER, [29]) and the Packed Encoding Rules (PER, [30]). The XER encoding produces human- and machine-readable XML documents from in-memory data structures. BER defines a self-contained format that augments the actual data of a document with type and length information such that it can be decoded by a receiver that has no knowledge of the original ASN.1 document. Since XER and BER documents are quite verbose, PER was designed to yield a very effective and bit-length optimized encoding. Nowadays, ASN.1 has mostly been superseded by XML Schema technologies and as discussed in [31], it is not inherently well-suited for resource-constrained devices.

XML Schema [32], [33] as the latest and most generally accepted standard for the definition of data types commonly uses verbose, human-readable XML as payload format, which is unsuitable for the use in WSNs. Even so, approaches
enabling XML processing on sensor nodes exist. For instance, Buschmann et al. propose “<<ASTAX” [34] that allows for an event-driven programming of WSNs that resembles the Simple API for XML (SAX).

An in-depth overview of compact encodings (e.g., ITUs standard X.891 “Fast Infoset” [35] or XGrind [36]) for XML documents is given by Werner et al. [37]. The authors introduce a novel technique called Xenia that yields a very effective encoding of XML instance documents that surpasses previously available XML compressors in terms of encoding efficiency. By constructing a pushdown automaton from an XML Schema document, Xenia encodes an XML instance document as a path through the automaton, which can be encoded very efficiently. The key difference between Xenia and our approach (cf. Section IV) is that we do not compress XML documents but in-memory data structures. Xenia optimizes the encoding of the markup of XML instance documents and does not focus on an optimized encoding of character data. We perform both optimizations, which is beneficial to reduce the payload length of serialized in-memory data structures (for details, please refer to our previous work [31]).

E. Transport

The XML-based SOAP protocol [17] describes how messages, exchanged in a Web Service-based SOA, are encoded. Even though the serialization technique of SOAP messages is not fixed by the standard, serialization using XML is recommended and also used predominantly. To exchange SOAP messages, a Web Service transport protocol is required (see Figure 2). Nowadays, the HTTP protocol is the most used transport binding for Web Services. But the SOAP specification does not oblige the use of a specific transport protocol or transport binding. Instead, Part 1 of the SOAP specification (SOAP Protocol Binding Framework) describes the requirements of SOAP compatible transport protocols. Choosing the appropriate transport binding depends on the individual system requirements. In the following we describe the most important Web Service bindings and their applicability for sensor networks.

The HTTP (Hypertext Transfer Protocol, [38]) supports the two most important message exchange patterns (MEP) of SOAP (request-response and one-way) and is available in virtually every Enterprise IT environment. Even though, the HTTP Web Service transport binding can be used in an asynchronous way, where request and response messages are sent over different HTTP connections, usually HTTP is used synchronously. Werner et al. [37] measured the protocol overhead of HTTP on the application layer with 560 Bytes for a simple asynchronous SOAP request calling a Web Service method with no input and no output data.

FTP (File Transfer Protocol, [39]) is mentioned very often as alternative transport binding to HTTP, it is not often used in production systems. FTP itself only supports one-way as MEP. For FTP (and any other asynchronous transport binding) to support request-response, the use of WS-Addressing [40] is necessary. It extends the SOAP header and allows a correlation of request and response messages. Werner et al. report an overhead of 576 Bytes for a simple Web Service request for the FTP transport, which is similar to HTTP’s overhead.

In addition to HTTP and FTP, the W3C has specified a SOAP-over-email binding [41]. Just as FTP, e-mail communication is asynchronous. Unlike the HTTP- or FTP-binding, the e-mail binding does not specify a transport protocol. Emails can be exchanged using different protocols, but typically SMTP (Simple Mail Transfer Protocol, [42]) is used to send SOAP messages. Compared to HTTP or FTP, the communication overhead of 2,535 Bytes is by a factor of 4.5 higher. Just like Microsoft Message Queuing or Java Message Service (see below), the email binding allows a one-to-many communication.

The Microsoft Message Queuing (MSMQ) and the Java Message Service (JMS) protocols realize an asynchronous message exchange following the store-and-forward-mechanism. As SMTP, JMS and MSMQ store messages sent by a Web Service client in a message queue. Thus, Web Service client and server do not need to be online at the same time. Next to point-to-point communication, MSMQ and JMS enable publisher/subscriber communication by allowing messages receivers to register for certain message types. But other than SMTP, which is an open standard and widely used in the internet, MSMQ and JMS are proprietary technologies. Thus, their use outside enterprise boundaries is very limited. The communication overhead of MSMQ on the application layer is 2,959 Bytes (from Werner et al.).

So far, all the protocols described above are application-layer protocols. The overhead of HTTP (560 Bytes) or FTP (576 Bytes) is too large for the wireless interface of typical sensor network platforms. To exchange messages of application layer protocols, underlying transport protocols are required, too. Thus, additional communication overhead is generated. An approach to reduce the communication overhead is to exchange SOAP messages without the use of application layer protocols. Instead, SOAP messages are directly exchanged using TCP (Transmission Control Protocol, [43]) or UDP (User Datagram Protocol, [44]).

The SOAP-TCP binding enables the exchange of SOAP messages directly over TCP sockets. This binding is provided in a synchronous and asynchronous variant. While during a synchronous exchange both the request and response are sent over the same TCP connection, the TCP connection is closed right after the transfer of a request message in case of using the asynchronous variant. Responses to asynchronous requests are sent using a new TCP connection. The communication overhead of a simple SOAP message is 538 Bytes. Since the application layer protocols described above are TCP based, one has to add a TCP overhead to those measurements, too. But even though, these values cannot be compared directly with each other, the deduction that the overhead of TCP, HTTP, FTP, SMTP, MSMQ, and JMS is too resource demanding for typical sensor network platforms, can be made. Furthermore, TCP has large requirements to link stability. But usually, wireless sensor networks are characterized by unstable as well
as unidirectional links [45], which complicate the applicability of TCP in sensor networks.

Combs et al. [46] present a SOAP binding for UDP. It provides an IP unicast, multicast and broadcast connection on the IP layer with no additional communication overhead. Compared to TCP or TCP-based Web Service bindings, UDP does not provide mechanisms to control the data flow. Furthermore, it supports only a maximum message size of 64 kBytes. But with only 16 Byte overhead, UDP is significantly less resource-consuming than TCP.

In comparison to Web Service transport protocols of the application layer, TCP and UDP are not capable of addressing different Web Services by themselves. One can only address a specific application on the target computer. But one application can host different Web Services. Thus, the use of WS-Addressing is required, which causes larger SOAP messages and neutralizes at least parts of the reduced overhead compared to the use of an application layer transport binding.

Werner et al. present PURE [47] as a lightweight Web Service transport binding for resource-constrained devices. PURE is UDP based but avoids the disadvantages of UDP. First, UDP is extended with a message flow control. Second, PURE provides a packet fragmentation mechanism to eliminate UDP’s maximum packet size of 64 kBytes and enables unlimited packet sizes. The communication overhead of PURE is 66 Bytes. Considering that UDP’s maximum message size of 64 kBytes cannot be handled by most sensor network platforms anyway, PURE should be preferred over UDP in WSNs if reliable messaging is needed.

All protocols described above require IP on the network layer. Considering the integration of classical IP based networks and resource-constrained sensor networks, there are two oppositional opinions in the research community. One group is convinced that neither IPv4 nor IPv6 is applicable for sensor networks [48]. But on the other hand, more and more researching groups have published results and implementations of software stacks showing that optimized IP stacks can fulfill the restrictions of sensor networks in terms of limited memory and computation power [49].

With µIP [50], Dunkels et al. demonstrated the feasibility of an RFC-compliant IPv4 stack for 8-bit microcontrollers. Analog to µIP, Durvy et al. [48] presented uIPv6 as small IPv6-ready stack for resource-constrained devices. µIPv6, which is tightly coupled to UDP and TCP, has a code size of 11.5 kBytes and requires less than 2 kBytes of RAM. Another approach of transferring IPv6 into sensor networks is 6LoWPAN, 6LoWPAN is an IETF standard [51] that specifies how IPv6 datagrams are exchanged over an IEEE 802.15.4 wireless interface. It supports header compression as well as datagram fragmentation.

With the solutions on the network, transport and application layer described above, it is possible to realize a SOA on less resource-constrained sensor network platforms. For instance, Priyantha et al. [32] describe a Web Service-based approach for building up a dynamic sensor network where additional nodes can be added after the initial setup of the deployment.

The main research challenges addressed by the authors are the support of duty cycled nodes, message serialization and processing as well as energy costs. This approach is based on standard technologies such as IPv6, 6LoWPAN, TCP, and HTTP. Considering the message serialization and transport, Web Service messages are exchanged using (TCP based) HTTP. As far as possible, they avoid using complex SOAP messages. Instead, if the Web Service messages do not contain complex data structures, simple URL encoded messages are exchanged to reduce the message size. But regardless of the overhead of SOAP or URL encoded Web Service calls, the overhead of TCP and HTTP is by far too large to be applicable in most sensor network deployments.

Amundson et al. [10] present a SOA-based approach for sensor networks. It enables a service-oriented communication within the sensor network. Furthermore, applications respectively services executed on sensor nodes can call Web Services hosted on Enterprise IT servers. The inter-service-communication within the sensor network is not based on Web Service technology. They have developed a proprietary middleware solution enabling remote service calls and additional services such as service discovery. To enable sensor nodes calling standard Web Services of the Enterprise IT, a gateway is needed which converts between the proprietary middleware message format and standard Web Service messages.

IV. SEAMLESS WEB SERVICE BASED INTEGRATION OF SENSOR NETWORKS AND ENTERPRISE IT SYSTEMS

In the enterprise IT context, there are a lot of technologies for the implementation of a Web Service based SOA (including BPEL). Our main research focus is the integration of mobile and wireless sensor networks into enterprise IT systems with Web Service technologies for sensor networks. From the technical as well as from the business perspective, to realize a seamless integration of sensor networks and enterprise IT services, it is necessary to extend today’s SOA technologies rather than developing new solutions designed especially for sensor networks. In the previous sections, we pointed out the research challenges, which have to be solved on different layers of the Web Service technology stack to apply the Web Service technology to sensor networks. Furthermore, we presented a taxonomy of Web Service related technologies.
as well as sensor network related technologies and showed pros and cons of applying these technologies to realize a Web Service based SOA for sensor networks.

In this section we present our approach that enables a seamless Web Service-based SOA, which complies with existing Web Service standards. As Figure 4 shows, it allows the development of services in the WSN, the Enterprise IT, and user domain by using exactly the same Web Service technology to communicate with services in the same environment as well as for the inter-domain communication. Thus, it is possible to design business processes on top of this architecture using standard business process modeling technologies such as BPEL, ebXML, or BPEL4People (WS-BPEL Extension for People, [53]). The use of standard business process modeling technologies, which offer a large graphical tool support, allows business processes to be flexibly designed even by non-IT personnel. Figure 5 describes the architecture of a typical application system based on our Web Service technology stack for sensor networks approach and Figure 6 describes how the standard Web Service technology stack has to be extended to be applicable in sensor networks and enterprise IT environments (elements with white text on dark background).

A. Messaging

To realize this, we need to extend the standard technology stack (see Figure 2) on the message as well as on the transport level (see Figure 6). The first challenge is the serialization of Web Service messages. Usually, they are encoded as XML/SOAP messages. Due to the resource constraints, the bulky XML documents cannot be handled by sensor nodes for two reasons: bandwidth and memory. As the bandwidth and maximum messages sizes in WSNs are too small, one has to compress SOAP/XML messages, e.g., by using GZIP [54]. This would solve the problem of the limited bandwidth, but to process the GZIP compressed XML message on a sensor node, one still has to decompress the message in memory.

To address both challenges, we use the Fabric/Microfibre-framework [31] to compress SOAP/XML messages. Just as with SOAP, messages and operations are described using XML Schema and WSDL. Using these formal descriptions, source code is generated that (de-)serializes messages from and to very compact binary representations or standard XML documents. In contrast to the above-mentioned GZIP compression, messages are not represented as XML documents in mem-
ory but in standard constructs of the programming language such as structs in C or classes in C++ or Java and hence significantly reduces the memory footprint. The achievable compression rate exceeds that of generic GZIP compressors and even that of existing schema-driven approaches [31].

As the SOAP standard does not require a specific serialization method, the use of Fabric/Microfibre-encoded messages fully conforms to the Web Service standards. As a result, SOAP messages can either be serialized by using Fabric/Microfibre or Fabric/Microfibre can be used to transparently (de-) compress XML serialized SOAP messages to compact binary SOAP messages, e.g., at a gateway.

B. Transport

The second challenge to make Web Service technology seamlessly applicable for sensor networks is the transport of Web Service messages. The right part of Figure 5 shows a typical application where two different companies interacting using four different patterns:

1) Sensor nodes which are part of the same sensor network are communicating with each other.
2) Enterprise IT services of one company are communicating with enterprise IT services of the same as well as of another company.
3) Sensor nodes are communicating with enterprise IT services of the same as well as of another company.
4) Sensor nodes of one sensor network communicate with sensor nodes of another sensor network. Even though the other sensor network belongs to another company and uses incompatible radio interfaces.

To support these four communication patterns using a Web Service message transport, a transport binding, which supports arbitrary underlying transport protocols, is required. As TCP/IP and UDP/IP is omnipresent in Enterprise IT environments, this is easily accomplished. However, as discussed in Section III, this assumption does not hold for the most WSN platforms. In addition, the routing protocol of WSN applications is often very application-specific, e.g., because of duty-cycling, clustering, etc.

To address exactly these issues, we have designed a Web Service transport binding for WSNs (WSNTB) capable of using an arbitrary transport protocol and routing protocol. For instance, within an enterprise IT environment, WSNTB could use TCP/IP or UDP/IP, while any available protocol may be used in a WSN. In the case of message exchange from the Enterprise IT and a sensor node and vice versa, a lightweight gateway converts the transport protocol from TCP/IP or UDP/IP to the specific sensor network routing protocol. This may of course also be an IP-based protocol such as μIPv6 or 6LoWPAN, however, these also require the presence of some kind of gateway to be present. To support communication from a sensor node located at company A to a sensor node located at company B, messages are routed via the respective gateways. Between these gateways, a classical IP-based networks (e.g., the Internet) is used.

However, all communication is based on WSNTB and hence, the addressing of services is transparent for the application developer. WSNTB has a consistent addressing scheme following the URL notation (e.g., wsn://host/service). A developer only needs to know such an endpoint of a service. But he does not have to care if the destination service is deployed on a sensor node or an enterprise server. All communication details are hidden to the developer and are handled by WSNTB. As an optional feature, we envision a header compression mechanism to avoid these lengthy URLs when messages are entering the sensor network to further reduce the packet sizes.

C. Converting Web Service Protocols

Our extensions of the Web Service technology stack are absolutely compliant to the relating Web Service standards. But of course, when using some kind of standard, every communication partner has to use this standard as well. The HTTP- or SMTP-transport bindings as well as XML on the message layer are widely used and hence, one can assume that they are available in virtually every enterprise environment. The use of another transport binding or message serialization therefore limits the possible communication partners to those supporting the same technologies (e.g., inside the same enterprise or partner companies). For any other communication partner, a protocol conversion mechanism is required.

In Enterprise IT environments, such a protocol converter is realized within an enterprise service bus (ESB). A major task of an ESB is to allow communication between services, which use different message or transport protocols. So, to also enable the communication with services not using WSNTB or Fabric/Microfibre, we currently implement a gateway, which converts between commonly used protocols and Fabric/Microfibre serialized SOAP over WSNTB (see Figure 5). For the production environment, this gateway component will be integrated into the open source Enterprise Service Bus Open ESB [55].

V. CONCLUSION AND FUTURE WORK

Two major but so far unfortunately separate trends of the Future Internet are the integration of all kinds of resource constrained devices and a consequent service orientation. In this paper we show how wireless sensor networks can be fully and seamlessly integrated into SOA environments based on standard-compliant Web Service technology despite their limited resources.

After defining the research challenges, we present a taxonomy of existing approaches of all layers of the standard Web Service technology stack. As first step to realize the integration of sensor networks and enterprise IT systems, the message and transport layer of the Web Service technology stack require extensions. With WSNTB we propose a Web Service transport binding, which seamlessly exchanges Fabric/Microfibre serialized SOAP messages in wireless sensor networks as well as in classical IP based networks.

The aspect of Quality of Service (QoS) was not discussed in this paper. But of course, in enterprise applications systems
a certain level of QoS must be provided. E.g., no user would communicate credit card information unencrypted. A lot of business processes run in the context of transactions. Transferring technologies realizing these QoS concepts to sensor networks will be future research challenges for the Web Service oriented sensor network community.

REFERENCES

Design and Implementation of a Gateway for Web-based Interaction and Management of Embedded Devices

Vlad Trifa*, Samuel Wieland*, Dominique Guinard*†, and Thomas Michael Bohnert†
*Institute for Pervasive Computing, ETH Zurich
†SAP Research CEC Zurich
8008 Zurich, Switzerland
Email: vlad.trifa@ieee.org

Abstract—Wireless Sensor Networks provide unprecedented possibilities for monitoring and interacting with the real-world. Unfortunately, the lack of open and simple standards for ad-hoc collaboration between heterogeneous embedded devices makes it difficult to build large-scale deployments; every particular application requires complex integration work, and therefore technical expertise, effort and time. Inspired by the success of Web 2.0 mashups, we propose a similar lightweight approach for interacting with networked devices. In particular, we describe a gateway architecture that enables to access sensor nodes through a RESTful interface. With this approach, interacting with a sensor node becomes as easy as typing a URI in a Web browser. By reusing the architectural principles of the modern Web, we show how one can built a loosely coupled infrastructure for the Web of Things that scales well and extends the current Web to the real world.

I. INTRODUCTION

Although the field of wireless sensor networks (WSN) is still in its infancy, companies that sell embedded sensing systems are already flourishing (Sentilla, Arch Rock, Streetline, etc.). These devices provide unprecedented possibilities for monitoring and interacting with the real-world and could be an invaluable help in many disciplines. Unfortunately, most sensor network projects have focused on building vertical solutions designed for very specific applications that run as isolated, small scale testbeds. As a consequence, the lack of commonly agreed standards for sensor networks has resulted in a wide variety of hardware and software platforms that are usually incompatible. Without a set of simple and open standards for ad-hoc networking and interaction with embedded devices, building and maintaining large-scale sensing applications on top of embedded devices requires extensive effort and expert knowledge, in particular when devices from different constructors are used. To hide the complexity and heterogeneity of devices and network protocols used, various middlewares for sensor networks have been proposed [1]. Unfortunately, most existing approaches are too complex for non-experts, and also are based on tightly coupled components, which strongly affects the scalability and evolvability of the whole system. As a consequence, a worldwide Sensor Web (also called the Internet of Things) where billions of “smart” objects are shared and can be easily reused, still remains unfeasible.

As demonstrated by the success of the Web, loosely coupled approaches posses high scalability and robustness - which are fundamental properties for building a worldwide network of devices. Furthermore, the real value of such applications comes from the sharing and integration of data among heterogenous devices. Based on these considerations, we describe in this article a generic and easy to use middleware to enable ad-hoc interaction between embedded devices. Our key requirements are to maximize reuse and sharing of embedded devices, while minimizing the time needed for fast prototyping applications that run on top of physical sensors. Because the number of embedded devices with direct Internet connectivity is rapidly growing, we propose to fully leverage the existing and ubiquitous Web standards to build a middleware for embedded devices.

Our approach is based on smart gateways, which are lightweight and extensible software components that enable Web-based interactions with all kinds of embedded devices. In addition, gateways can be linked together to form hierarchical trees that can be further mapped to physical locations. In this manner our approach can highly scale, and at the same time support location-based services. The role of gateways is not to replace existing sensor networks, but to facilitate finding and reusing of shared devices using standard Web technologies. From the perspective of a web programmer, devices become web resources that can be addressed and used to build mashups. More experienced programmers can change the driver implementation that controls the communication between the sensor network and the gateway to fit their particular needs. For WSN developers, the gateway simplifies greatly the process to export data and functionality of the WSN to be provided to end users on the web. Using our gateway system do not require to change existing deployments, but only to implement a specific driver into the gateway to interact with the sensor network.

For certain applications tight coupling using proprietary solutions remains the most desirable choice for building high-performance systems with real-time requirements (as for exam-
ple in the industrial automation or banking domain). Specific tight-end coupling can be used in back-end for such very particular tasks (based on proprietary solutions). These systems can expose their functionality in a high-level abstraction that can be accessible from the web in a plug-and-play manner so that it can be used directly over HTTP. However, much simpler loosely coupled approaches are to be preferred for tasks with more modest requirements (which represent most use cases for data monitoring and home applications), because of their inherent flexibility and intuitive use.

II. RELATED WORK

The idea of linking physical objects with the Web is not new, and early approaches used physical tokens (such as bar codes or RFID tags) to retrieve information about objects they were attached to [2], [3]. For example, in the Cooltown project [4] each thing, place, and person have an associated Web page with information about them. Shaman was an early gateway system that enabled low-power devices to be part of wider networks [5]. With advances in computing technology, tiny Web servers could be embedded in most devices [6]. The idea of each device having its own Web page is appealing because device pages could be indexed, searched, and accessed by search engines, and this directly from a Web browser. However, static indexing of mobile devices is not possible when new devices appear and disappear continuously. Besides, the goal of earlier work in Web-enabled devices was to provide an online representation of real things for humans (real-time status displayed on a HTML page), but no attempts to seamlessly integrate devices into the Web as proactive units nor enabling sharing and reuse of device-level functionalities has been mentioned.

Many technologies for building distributed applications on top of heterogeneous devices have been proposed. Among them, the now classical systems are CORBA, JINI, or RMI. JXTA [7] is a set of open protocols for allowing devices to collaborate in a peer-to-peer fashion. JXTA was among the first real attempt to bridge physical objects in the world with the Internet. More recently, Web services have also been used to interconnect devices on top of standard Web protocols [8]. However, these approaches are based on tightly coupled solutions, where each element had full knowledge about the other peers and the functions they offered. Unfortunately, such solutions are too rigid to deal efficiently with the constraints and requirements of mobile embedded devices, in particular for ad-hoc interaction with new devices with unknown properties.

Several systems for integration of sensor systems with the Internet have been proposed [9], [10], [11]. SenseWeb [12] is a platform for people to share their sensory readings using Web services to transmit data onto a central server. Pachube [13] offers a similar community Web site for people to share their sensors and uses more open data formats. Unfortunately, these approaches are based on a centralized repository and devices need to be registered before they can publish data, thus are not sufficiently scalable. Prehofer et al. [14] recently proposed a Web-based middleware that is similar to our approach, however, they used the Internet only as a transport protocol, and no references to use a fully Web-like approach has been mentioned. Also, an interesting approach to use the web architecture and the semantic Web technologies can be found in [15].

However, most of the existing Web-based approaches use HTTP only to transport data between devices, whereas HTTP is in fact an application protocol. Projects that specifically focus on re-using the founding principles of the Web as an application protocol are still lacking. As pointed out in [16], creation of devices that are Web-enabled by design would facilitate the integration of physical devices with other content on the Web, in which case there would be no need for any additional API or descriptions of resource/function. The approach found in [17] is similar to ours, but focuses mainly on the discovery of devices and a more systematic approach and system evaluation is lacking.

III. INFRASTRUCTURE FOR THE WEB OF THINGS

Even though more and more embedded devices are being connected to the Internet [6], [18], a common ground to enable them to communicate using a uniform abstraction is still lacking. As illustrated by the growing popularity of open source communities and of the do-it-yourself technology, it is very likely that billions of sensors and physical objects throughout the world could be soon available on the Internet. The central research question remains - how can one build a (globally) scalable network of physical devices where these devices can interact together without any a priori knowledge about each other? As many appliances have built-in Web servers, and Web access is available practically everywhere with mobile phones, we propose to reuse the core principles of the modern Web architecture for sensor networks to take advantage of the scalability, reliability and “mashupability” properties of the Web. These principles are summarized under the REST architectural style as described in [19].

The success of todays Web can be explained to a great deal to its simplicity and openness. Only a few easy guidelines describe how Web components should be developed and integrated into the Web - and following them leads to so-called RESTful applications:

- Resources are uniquely addressable through URIs.
- The format and type of the resource representation is described through MIME-types which give the communication partner the possibility to chose the most suitable resource representation available (example: language aware websites).
- Resources can be accessed/modified with HTTP instructions PUT, POST, GET, DELETE that are comparable to traditional database operations. All the instructions are self explaining by their name which leads to a much better understanding of Web interfaces.

-
IV. Gateway System Design

The gateway architecture was designed with three main goals in mind: simplicity, extensibility and modularity. Extensibility enables users to extend and customize easily the gateway to fit their needs and their setup. Modularity so that internal components of the gateway can interact only through small interfaces, thus allowing the evolution and exchange of individual parts of the system. Our gateway implementation is written mainly in Java with several driver components in C/C++ or Python. The architecture is composed out of three major layers - the presentation layer (IV-B), the control layer (IV-C) and the device abstraction layer (IV-D) - each responsible for a well defined set of tasks. Figure 1 shows a high level overview of the gateway.

![Gateway Architecture Diagram](Image)

**Fig. 1.** High level overview of the gateway architecture. The illustration shows the different Web interfaces offered to external clients. Each plugin is represented by a box, and new plugins can be actually added and removed at runtime. The core module that takes care of starting and stopping all the other modules is not shown on this figure.

A. Core

The core specifies the “kernel” of the gateway and is in charge of managing the components of the architecture, and also providing utility classes for all of them. To ensure that there is only one core available at runtime, the “Singleton-pattern” [20] is used. At startup time, configuration files are loaded, the HTTP server for the presentation layer (IV-B) is started and the plugins for the control layer (IV-C) are initialized. At shutdown the core makes sure that the current configuration of the gateway gets stored back to the configuration files and that all the resources occupied are freed again.

B. Presentation

The presentation layer makes the gateway components accessible to the outside world. It is a thin layer on top of the control layer (IV-C) managing requests from clients through a REST interface. We have used the Restlet framework\(^1\) as HTTP server in order to model the most important HTTP operations (GET, POST, PUT, DELETE).

All the resources on the gateway can currently be retrieved in four different formats (XML, RDF, HTML and plain text) allowing clients to choose their preferred format. In HTTP requests, the MIME type is used to specify the desired format. On the server side, in a first phase an XML is generated and then translated with XSLT stylesheets on the fly. This allows machines to use semantically enriched RDF formats whereas humans can retrieve an HTML representation. In addition new formats can be easily added by using XSLT stylesheets.

In order to make the devices attached to the gateway accessible through the Web, a mapping from device names to URI is performed by the presentation layer. A device with the name “sensor1” will be mapped to “/sensor1”. This allows users to browse dynamically the device list. Requests from the web onto such a mapping will be redirected to the responsible device driver that then is responsible to handle the request accordingly (IV-D).

C. Control

The control layer is composed of several independent components called plugins. A plugin is a software component that is loaded at startup of the gateway through the core (IV-A). Users can write their own plugins and place them (packed as a jar file) into the classpath of the gateway. The jar file has to contain a “marker”-file `props/plugins.xml` that contains a descriptor for the plugin with the mandatory fields Version, ID, and Dependency (IDs of plugins that need to be loaded before this plugin). The “marker” contains the class name that will be used later for the class-loading. Plugins are allowed to depend on other plugins (example: the eventing plugin depends on the device management plugin). To keep a loose coupling between plugins, a lightweight synchronization mechanism is implemented with the “observer pattern” [20]. The observer (plugin depending on some other plugin) will be registered on the observable. As soon as the observable changes its state, an internal synchronization method is invoked informing all the observers about the change (example: the devices plugin calls this method whenever a device is removed or added).

1) **Device management plugin**: The Device Management plugin maintains a high level view on devices registered at the gateway by using the device abstraction (IV-D). New device drivers are loaded through dynamic class loading. By sending a PUT request to the device management with the parameters `class` (fully qualified name of the class implementing the device driver) and a unique device identifier `ID` new instances of the driver instances can be invoked. Other plugins (or other layers within the architecture) can query and access the devices currently registered at the devices plugin by using the unique id.

On each device driver, a “heartbeat” protocol is installed that requires the device to keep its status “alive”. This means that at regular intervals, a watch-dog is invoked and iterates over all the drivers and invokes the method `isAlive()`: boolean. If a driver does not respond with `true`, the driver gets removed from the device list. Note that the implementation does not

---

\(^1\) [http://www.restlet.org](http://www.restlet.org)
poll the real device for aliveness - the aliveness test within the
driver towards the physical device is usually performed using
an optimized manner depending on the device.

2) **Eventing plugin:** Many sensors read their state in a
regular interval and a programmer usually has to check
periodically whether this state has changed or not. When
using sensors over the Internet, polling is inefficient and
creates unnecessary load that can be avoided by using an
asynchronous publish/subscribe model.

From the clients perspective, this is a simple **POST**-request to
the event registration URI with three mandatory parameters -
the lesstime to specifying how long the registration shall be
valid, the keyword specifying the type of event the registration
is for and the callback giving an address where to deliver the
events (see Figure 2). Consider the tree structure from Figure
3. The client registers at the top node (floor1) for a “fire”-event
triggered by any of the sensors in the subtree of the “floor1”-
gateway. As soon as a “fire”-event is triggered on the gateway,
the client will be notified about the event through the callback
address provided in the registration request. In our case, the
“floor1”-gateway is also connected to other gateways in its
subtrees. It therefore has to register on these gateways as well
for fire-events. This registration works in exactly the same
manner as with the client registration, but this time with the
“floor1”-gateway as client on the different “room”-gateways.

**POST** /_eventing/registration HTTP/1.1
Host: ip_of_the_registering_client
lesetime=6000
callback=client_callback_address
keyword=fire

Fig. 2. Example for a client event registration. The registration is valid
for 6 seconds (6000ms). Events of type “fire” will be sent to the callback
"client_callback_address".

Fig. 3. Sample gateway hierarchy with a gateway at the top of the tree
(floor1) with two gateways in the subtree (room1 and room2). The solid arrows
indicate the path the registration message takes through the gateway. The
pointed arrows indicate the path the event message follow.

In the current implementation of the gateway, we use
this simple eventing approach. However, more advanced
approaches would be more appropriate for larger scale imple-
mentations. We are currently investigating message broker
methods such as MQTT² or XMPP³.

### D. Device Abstraction

Like in almost any modern operating system, the gateway
provides an abstraction for devices. For higher level applica-
tions, any type of device looks the same, even if the underlying
implementation differs. The mechanism is comparable to the
one used in Section IV-C.

The device abstraction is illustrated on the left part of Figure 1.
Different specialized drivers are used to communicate through
their "proprietary" protocol with the respective physical device
(example: SunspotDriver communicates through ZigBee with
the SunSpot). However, from a higher level perspective all
the drivers implement the abstract class **Device** allowing the
device management plugin to treat all the devices in the same
manner.

For devices that are already Web-enabled (that is they support
HTTP over TCP/IP), the driver implementation can simply
forward requests from the presentation layer (IV-B) to the
physical device. However, when the device is not Web-enabled
(e.g. sunspots do not have an IP stack), the driver is responsible
to translate the web request into a protocol understood by the
physical device.

A crucial feature of a device driver is the capability to act
as a resource representation (or proxy) for the underlying
physical device. Consider a temperature sensor changing its
temperature seldomly. Instead of polling the device every time
a client requests the temperature, the driver can store the
temperature and return the value directly, thus minimizing the
actual communication with devices. This caching mechanism
is very useful for shared access to quasi real-time sensor data
collected with low-power devices.

Along with the eventing mechanism (Section IV-C2) and with
the fact that all devices are accessible as Web resources, it
is easy to compose higher level devices by combining devices
from lower level devices (for example the “RoomState” virtual
resource in Section V, which is actually a combination of
physical sensors).

### V. EXPERIMENTAL DESIGN

To illustrate the different properties of the proposed gateway
architecture, we have implemented a simple prototype scenario
illustrated in Figure 4. The gateway was installed on a laptop
computer (2 x 1.6GHz, 2GB RAM), where we attached a
SunSpot base station for ZigBee communication and a Tikitag
RFID reader⁴ to read the Tikitag RFID tags. As SunSpots per
se are not IP capable, a reverse proxy (described in [21]) is
used to multiplex ZigBee communication streams so that the
SunSpots can access Web content directly through HTTP. The
virtual RoomState device simulates a temperature regulator
of the room. This regulator can increase or decrease the
temperature, and is capable of returning the current value (both
directly using REST).

²[http://mqtt.org/](http://mqtt.org/)
The RoomState device has registered for events sent by the Tikitag reader. By placing different Tikitags onto the RFID reader, users can dynamically adjust the value of the roomstate (e.g. using tag A increases the temperature on the RoomState device by one degree, whereas tag B decreases the value accordingly).

The SunSpot periodically polls the RoomState device on the gateway using a GET on the URI of the RoomState resource to read the current value (the temperature of the room). The value is then displayed using different LED colors (e.g., blue when the temperature is below 10 degrees, green when between 10 and 25 degrees and red when over 25 degrees). Additionally, the SunSpot can also be used as an input to control the room temperature, for example by using the acceleration sensor onboard (shaking the sunspot for 5 seconds will issue a POST request on the RoomState resource). This simple system illustrates a trivial mashup built on top of completely different devices, where their interaction is only defined by calling different URIs.

VI. SYSTEM PERFORMANCE EVALUATION

Based on the prototype we described in Section V, in this section we evaluate a few interesting aspects with a larger evaluation setup. We have implemented the gateway on two different test platforms.

- Two computers both running gentoo gnu linux with sun-jre-1.5.0.17 linked by a 1Gbit ethernet. The gateway was installed on the test server (1.1GHz, 2GB RAM, 1Gbit NIC). The test client (2 x 2.13GHz, 8GB RAM, 1Gbit NIC) simulated several clients “calling” the server.
- One NSLU running debian gnu linux with sun-jre-1.5.0 as test server (133MHz, 16MB RAM, 100Mbit NIC) and a test client (2 x 2.13GHz, 8GB RAM, 1Gbit NIC).

A. Subscriber Churn

In this test, we evaluate the gateway eventing mechanism under high churn (meaning that many clients are registering and leaving). The parameters for testbed 1 (Figure 5) have been set to 100, 500 and 1000 simultaneous clients, for testbed 2 (Figure 6) to 20, 40 and 100.

When the gateway is installed on the NSLU (setup 2), the results might indicate a performance problem. To verify this assumption, we must run the tests also on different hardware with comparable CPU power. On the faster server, however, the test results indicate that the gateway scales nicely with a large number of simultaneous clients. With a moderate increase of computational power (CPU and memory), a much larger number of clients could be supported.

B. Caching

To speed up the response time when delivering a page that has to be generated, a caching mechanism has been used within the gateway. As long as the representation did not change, the generated representation will be cached and the cache will be returned upon subsequent requests. In testbed 1 (Figure 7) and testbed 2 (Figure 8) 150 GET requests have been performed sequentially.

In both testbeds caching reduced response time significantly.

VII. DISCUSSIONS

In this article, we have described a lightweight gateway architecture to enable Web-based interaction with low-power devices that do not have direct connections with IP-based networks. As the gateways can be easily extended to incorporate new devices, users can easily create drivers for any embedded device or sensor network. This will enable to expose their functionality as URI-identified resources that are directly accessible on the Web, thus can be manipulated using HTTP.

We have shown that the gateway eventing infrastructure can scale enough to support many simultaneous clients with a reasonable speed. In most applications, room-level gateways will have only a few devices and users connected simultaneously, therefore are sufficient for most use cases where a delay of a few seconds is tolerable. The tests with the NSLU device have shown possible bottlenecks when dealing with constrained devices, so further work is required to reduce the computational requirements. Nevertheless, the results we have obtained are encouraging and the overhead introduced by our system when gateways are connected directly to the Internet are negligible. However, the overhead increases quickly when gateways are combined into complex trees, unless all the gateways in the system can be accessed directly from the Internet.

In addition, we have shown how caching of sensor values reduced response time significantly. This is especially true for reading data from physical devices that seldomly change their state. However, in the case where the cache entry is invalidated right after generation by a state change on the device, the cache maintenance overhead could degrade overall performance of the gateway. For most monitoring application where devices are shared publicly, data from devices can be accessed directly using the resource URI, while in fact the data is being generated by the gateways, and no real communication with the device is taking place. This process is fully transparent to Web clients, so this contributes highly
Web standards allow any device to speak the same language as other resources on the Web, making it much easier to integrate the real world with any other Web content, so that physical things can be bookmarked, browsed, googled, and used just like any other Web page. However, a resource-oriented approach shall not be religiously considered as the miracle solution for all problems. In particular, tightly coupled system that involve very specific functionalities and need high performance would still benefit more from the traditional RPC-based approaches. Nonetheless, Web 2.0 mashups have significantly lowered the entry barrier for the development of Web applications, which is now accessible to non-programmers. As demonstrated by the success of the Web, the development of a set of simple, reusable, and modular software components can greatly facilitate the integration of embedded devices within Web applications.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper we have shown how the Web can be used to build a network of heterogenous, yet interoperable, devices that can be found and used both by machines and humans. By using a resource-oriented approach and REST, instead of SOAP-based Web Services, we fully leverage the existing Web standards to increase the modularity and interoperability of the different components. This results in more flexible applications that scale well, but also that blend symbiotically with the existing Web without requiring any change. We have described a prototype of such a fully web-compliant architecture suited to the global scalability of the system, even when the gateway is running on a small device such as the NSLU.
for wireless sensor networks. We have used this prototype to illustrate how one can very easily combine real-time information from different physical devices without requiring any advanced knowledge in Web Services and complex middleware systems. Our proposal is to reuse the core principles of the Web (REST) as a basis for a lighter and standardized middleware for networking all kinds of devices. By considering sensor networks as distributed Web applications, one can shift from monolithic applications towards a fully decentralized and flexible solution. Such an open and standardized middleware would encourage widespread adoption by maximizing reuse and lowering the access barrier for people to use and develop applications on a much wider ecosystem of interconnected devices. In the long run, extending the Web to easily integrate physical devices will reshape the Internet into a multipurpose collection of physical and virtual resources that can be easily (re)combined at runtime to solve any task at hand.

REFERENCES


Abstract—Prior to its deployment a Wireless Sensor Networks (WSN) and especially the driving software need to be configured with some set of appropriate parameter values to optimize its operational readiness, the efficiency and the accuracy of the system. To avoid expensive large scale real world test deployments the search for such a configuration can be supported by simulation. This process is effective only if the simulation submodels are chosen and parameterized in a way that they are capable to imitate the intended operations environment in a reasonably accurate way. Hence the choice of configuration parameter values is an essential part of software engineering for WSNs.

In this contribution we advocate a course of action that enables us to find appropriate configuration values for both, simulation models and WSN specific protocols and applications. We present the ESAC-toolkit (Evolutionary Simulation and Application Configurator) that facilitates evolutionary algorithms for configuration support and showcase its application by two examples—one leveraging real world data for simulation model configuration and the configuration of a routing protocol depending on the previous example.

Index Terms—configuration, simulation, evolutionary algorithm, WSN

I. INTRODUCTION

In many cases, developers of Wireless Sensor Network (WSN) systems use discrete-event network simulation to study the effects of various implementation/deployment choices: radio transmitter power, WSN node count, routing algorithm and other parameters may have great influence on the global WSN behavior, and it is worthwhile to explore different settings. Software developers find network simulation especially useful during the integration test phase of the software development cycle: It may help them to ensure that the WSN nodes cooperate with each other as intended. Simulation is chosen as “tool of the craft” because an experimental full deployment of a WSN to its operations area is often prohibitively expensive and time-consuming, and thus developers strive to replace it by simulation whenever possible.

In order to produce useful simulation results, it is crucial to select appropriate simulation models, e.g. in terms of radio or mobility models, but also to configure these models appropriately. This means to find configuration parameter values—or a “set-up” instance consisting of parameter values for such models—that make the simulation resemble the intended real world deployment as accurate as possible in terms of relevant characteristics like e.g. radio connectivity or the distribution of nodes across the operations area. These characteristics imply the notion of “good” and “bad” set-up instances. It should be mentioned that even very sophisticated and complex models are somewhat useless if accompanied by a bad set-up instance and may even perform worse in terms of accuracy than simple models accompanied by good set-up instances.

Supposed that we have an accordingly accurate simulation environment with good model set-up instances, we may start to configure WSN related protocols or applications on this basis. The notion of a “good” set-up instance for protocols and applications is implied by user-defined specifications for the WSN, which depend on the users needs or expectations. It is again crucial to choose a good set-up instance to ensure the operational readiness and effective operation of the WSN finally deployed in the real world.

The process of configuration parameter value selection, especially the choice of simulation model related set-up instances is relatively often based on expert knowledge or estimations and handled as a minor matter besides other issues. Thus e.g. hardware specifications of some radio are consulted without questioning if these specifications are applicable unmodified for the selected models. Application related parameter values are typically inspected in more detail, e.g. by manually evaluating simulation executions according to certain performance metric while simply iterating some different configuration parameters values.

In this contribution we suppose that parameter value selection is a complementary step to model selection and as such a crucial step regarding software engineering for WSNs. Therefore it needs to be approached in a methodical way. This is supported by the ESAC-toolkit (Evolutionary Simulation and Application Configurator) as presented in the following. This toolkit provides a set of building blocks to find near optimal set-up instances for nearly arbitrary simulation models, protocols or applications. The proposed toolkit and the related process are particularly based on the simulation environment itself and evolutionary algorithms (EA). Even though EAs
and especially genetic algorithms have been applied to similar problems before (see “related work” section [VIII]), we set value on reusability and versatility of our approach which hence should be useful for a huge range of scenarios. Nevertheless a developer typically needs to fill in some (small) application specific implementations expressing e.g. the specific notion of “good”.

The toolkit will be motivated in this contribution by two examples: searching a set-up instance for a radio model on the basis of real world reference data and searching a set-up instance of an ad-hoc routing protocol according to some performance specifications.

The following section [II] provides a short overview of the buildings blocks of the ESAC-toolkit and their interaction. This is followed by a detailed discussion of these, i.e. the evolutionary approach (section [III]), the simulation environment (section [IV]), the fitness calculation (section [V]) and the visualization helpers (section [VI]). Section [VII] presents the two examples before the discussion of related work in [VIII] and some final remarks in [IX] conclude this contribution.

II. ESAC-OVERVIEW

As stated above the goal of the presented toolkit is to find good set-up instances for simulation models, protocols or applications—we will denote this specific part of the simulation the subject of optimization (SOO). Hence only parameters related to the SOO are candidates for being part of the set-up. Note that we distinguish the set-up instance which actually defines an assignment of values regarding a set-up. Furthermore the a set-up instance is part of a whole configuration instance (typically represented by a configuration file) of a simulation execution. We will discuss some rules of thumb and issues regarding the selection of parameters for optimization in [V-A1].

Supposed that we are able to conduct such simulation executions containing this SOO, we need to define the notion of good. We need some formal description to be able to rank the results of simulation executions in an automated way. This formal description will be called fitness function in the following. The fitness function may e.g. be related to application specific performance or QoS metrics. Based on the fitness function and resulting fitness values set-up instances can be compared.

To chose set-up instances one might simply select valid parameter values (valid according to the SOO) randomly, execute simulations configured with these set-up instances and compare these. Since the search space is often very large, this brute-force approach is in most cases not a viable approach—even if automated.

We propose to chose set-up instances in a ’smarter’ way based on previously evaluated set-up instances, i.e. on already gained experiences. This is actually the idea realized by evolutionary algorithms and supported by the ESAC-toolkit.

Besides components directly related to this search or optimization procedure, the ESAC-toolkit provides some helpers for visualization purposes easing the interpretation of the optimization process and its results.

This leads to an architecture consisting of four basic elements,

1) the evolutionary algorithm (EA, i.e. the optimizer),
2) the simulator,
3) the fitness adapter, coupling EA and simulator, and
4) the visualization helpers.

The fitness adapter combined with the simulator provide the fitness function taking one or more set-up instances and calculate according fitness values. The interrelation between the elements is depicted in figure [I]. Note that the EA and the simulator are meant to be existing implementations which are coupled by this approach. In the following we introduce the elements in detail.

III. EVOLUTIONARY ALGORITHMS

A. Sketch

Before introducing the concrete EA instances in use, we like to recapitulate some general properties of EAs for the reader which might be not familiar with this kind of approach. First of all EA is the name for a certain class of optimization algorithms, which are designed to find the global extremum of a function $f$ which is actually a black box to the EA. Even though we cannot provide a closed form analytic solution for $f$, it can be evaluated at any given point of its domain.

EAs are inspired by biological evolution where the population of a species adapts to its environment over the generations. The evolutionary algorithm manages a population of individuals where each individual represents a candidate solution, i.e. a specific set-up instance, in the search space as depicted in figure [2]. The search space contains at most all set-up instances which are valid with regard to the SOO. Typically the search space will be confined to a subset of all valid set-up instances—e.g. based on constraints of the scenario—to speedup the search. During the evaluation of the population a fitness-value is assigned to each individual. While fitter individuals survive, the others are discarded by selection. The next population is based on surviving individuals, but a slight variation is applied.

While there are many kinds of evolutionary algorithms, satisfying different needs, we focus on those capable of numerical optimizations, because parameters in the observed field are mostly numeric values. It follows that the representation of an individual is just a vector of real or integer numbers, forming
a set-up instance. Furthermore we assume only EAs which are capable of single objective optimization, which means that individuals are only ranked by a single value, the fitness which is a real number. Nevertheless we support the consolidation of multiple metrics to finally calculate a single fitness value representing multiple objectives. The fitness of an individual actually reflects how good the according individual or set-up instance performs regarding the given problem or reference. The fitness of an individual is independent of other individuals and same individuals have the same fitness regardless of the generation.

B. EAs in Use

A multitude of different evolutionary algorithms capable of numerical optimization are available. Common categories include:

- Genetic Algorithm (GA)
- Particle Swarm Optimization (PSO)
- Differential Evolution (DE)
- Evolution Strategy (ES)

As a starting point for the ESAC-toolkit we chose on the one hand to use Genetic Algorithms, since they have been proven to be useful and robust for a wide variety of problems. Although they are not very efficient in terms of the number of evaluations of the fitness function. This might be a considerable drawback since evaluating fitness function is linked to executing simulation runs in this case, resulting in significant processing time.

On the other hand we chose an example for Evolutionary Strategy, namely Covariance Matrix Adaption–Evolution Strategy (CMA-ES). As shown in [11] it finds a global extremum with less function evaluations than comparable algorithms. Although it copes very well with most of the common mathematical benchmark functions, it might be less robust compared to GAs, since it follows gradients of the studied function very stringently, increasing the possibility to get trapped in local extrema. Nevertheless it is easy to deploy because it has very few mandatory parameters. Furthermore it provides additional information like parameter correlations. These might be quite helpful when interpreting the results as will be shown in VII-A.

Genetic Algorithms

GAs are a group of algorithms which follow the evolution-ary idea very straightforward and without particularly sophisticated search strategies. It should be emphasized that there is not the one GA but a large variety of GAs based on different strategies. What they have in common is the general course of action as shown in figure 2. Although the selection and the variation step are provided with many different approaches. The latter is often split up in recombination of individuals which is often accompanied by some mutation.

GAs are very flexible but claim some creativity from the user to choose the appropriate strategies and configurations regarding selection, recombination and mutation to gain reasonable efficiency. For the ESAC-toolkit we implemented a GA based on the JGAP framework [11]. It utilizes an implementation of tournament selection [7] and single point crossover.

CMA-ES

Also starting from an initial population, CMA-ES samples a population of individuals around a mean \( m \in \mathbb{R}^n \) proportional to a multi-variate normal distribution \( \mathcal{N}(m, C) \), where \( C \in \mathbb{R}^{n \times n} \) is the covariance matrix determining the shape of the distribution. It then selects the best \( \mu \) individuals and computes the new mean \( m_{\text{new}} \) through weighted intermediate recombination. It also adapts the covariance matrix \( C \) in a way that increases the probability of successful steps to appear again [12]. Note that in contrast to GAs, CMA-ES does not sustain any individual of a population (not even the fittest) to be reapplied in the next population.

While not in an optimum the mean \( m \) follows the gradient of the function and the covariance matrix \( C \) stretches in the gradient direction (see figure 3). While in a local extremum the mean does not move much over some consecutive generations and the covariance matrix adapts to the inverse hessian matrix of the function [13]. By means of this adaption the iso-density lines of the normal distribution \( \mathcal{N}(m, C) \) align with the lines of equal function-value (see figure 3). This behavior enables CMA-ES to learn all pairwise dependencies between the parameters and the scaling of independent components.

Diagonal entries \( C_{ii} \) of the covariance matrix \( C \) contain the variance \( \text{Var}(x_i) \) of the corresponding parameter \( x_i \) showing how vague its current value is. Off-diagonal entries \( C_{ij} \)
contain the covariance \( \text{Cov}(x_i, x_j) \) between the corresponding parameters \( x_i \) and \( x_j \) that show interdependencies between those parameters. Correlations between parameters may be calculated based on \( C \) and can indicate shortcomings in the problem formulation, i.e. in the model or fitness function (see examples section below).

### IV. SIMULATION ENVIRONMENT

Since evolutionary approaches use characteristics of the evaluated function (especially gradients), results need to be comparable and therefore reproducible. Hence evaluating the fitness function with a certain set-up instance must always yield the same fitness value. That implies that each evaluation, first needs to be independent from other evaluations and second needs to provide uniqueness. Calculating fitness values from the output of an execution of e.g. a protocol and/or application instance, suggest for deterministic simulation, which can be provided by discrete event simulation. Note that emulation approaches e.g. influenced by real world aspects or operating system schedulers are possibly not appropriate for this task since they may not provide the uniqueness of results regarding some configuration.

This led us to chose a discrete event simulation environment, namely OMNeT++ \( \text{[2]} \) (currently version 3.4b2). Beside some valuable features like modular design and visualization capabilities OMNeT++ performs good in terms of runtime efficiency, e.g. compared to ns-2 \( \text{[23]} \). For the examples discussed below we used the INET framework \( \text{[2]} \) and the mobility model presented in \( \text{[16]} \) as extensions.

According to this decision we implemented the fitness adapter with regard to OMNeT++, even though other discrete event network simulators like e.g. ns-2/ns-3 would be appropriate too. Basically simulation runs need to be configurable via a textual configuration file. To adapt our approach to other simulators one needs to provide adaptations of the simulation preparation module and the parsers of the metrics processor which are discussed in the following sections.

### V. FITNESS ADAPTER

#### A. Conceptual Considerations

1) Set-up: As outlined in \( \text{[1]} \), only a small part of a simulation comprises the SOO and even not all configurable parameters of the SOO need to be part of the set-up. Some are clearly given by the application or operations area, others are obviously candidates for optimization and some deserve a closer look on if there is a need to consider them in the set-up.

Typical parameters which are given a priori are technical parameters like e.g. seeds for random number generators or parameters which are given by physical constraints, laws, standards or specifications. Thus the set of potential parameters regarding the SOO which are going to be part of the set-up can be narrowed significantly beforehand. The definition of the set-up should follow some basic rules or criteria:

- **Less parameters in the set-up lead to faster evaluations** since each parameter implies an additional dimension in the search space.
- **If it is possible to calculate a parameter from other parameters, it must not be part of the set-up.**
- **Depending on the fitness function some parameters influence the results in an obvious way.** If e.g. the number of nodes in a network would be part of the set-up and the fitness function rewards good connectivity, without taking into account the possible costs for additional nodes, optimizing the number of nodes may lead to very high node counts. Assumed that the fitness function was defined 'correctly', such parameters should be determined beforehand.
- **If the range of plausible values for a parameter is large and/or a good setting of it is unclear, this parameter should be included in the set-up.**
- **It is advisable to select the SOO and therefore the setup such that it does not comprise depending parts of a simulation, e.g. lower and higher layers, in the same optimization run but in separate runs (as it is exemplarily done in the examples section \( \text{[17]} \)).** Besides a possibly very large search space dependencies and according correlations between parameters may lead to unfeasible results.

For each parameter, which will be part of the set-up, a permitted value range needs to be specified. These ranges define the bounds of the search space and therefore influence the effort put into the optimization process. Hence these ranges should be chosen as narrow as possible but nevertheless carefully, to avoid missing out some possibly good set-up instances. The ranges are published to the optimizer which will avoid the selection of inappropriate values during the optimization process.

2) **Fitness Function:** ESAC provides the glue to couple a fitness function to an EA which is based on a discrete event simulation. Before we describe the fitness adaption, this section will give a conceptual overview on how fitness values for set-up instances are calculated.

As mentioned the fitness function defines what is supposed to be “good”. It maps a certain set-up instance \( x \) to a fitness value expressing the quality of \( x \):

\[
\text{f}_{\text{fit}} : \mathbb{R}^n \rightarrow \mathbb{R}, \quad x \mapsto f_{\text{fit}}(x) \quad (1)
\]

Although if we focus on the SOO, a simulation execution has various other parameters which will not be altered by the
optimization process. Hence the remaining parameters—not part of the set-up—need to be set to constant values. We call these remaining parameters the environment in the following, as they define the surrounding conditions for the optimization of the set-up.

The environment has a heavy impact to the optimization results since the optimization is run against this environment. Suppose for example that the environment specifies the number of nodes in a network and their motion pattern. Based on this environment a routing protocol should be configured. If the environment defines a sparse network with fast nodes this will probably yield to a dramatically different result than the case where the environment defines a dense network with slowly moving nodes. If the routing protocol should be applied for both cases—suppose e.g. a vehicular network covering suburban and downtown areas likewise—one needs to optimize against both. While this example is quite obvious more subtle situations are conceivable, e.g. the need for changing the seeds of random number generators. This problem may be denoted by the term of making the environment representative regarding the envisioned deployment.

To differentiate these ‘sub-environments’ composing the representative environment $E$ we call them scenarios $S_i$, i.e. $E = \{S_0, \ldots, S_{n-1}\}$ for $n$ scenarios. This implies for the fitness function to evaluate a set-up instance against all of the scenarios and merge the resulting fitness values.

Fitness computation relies on performance related metrics $m_0 \ldots m_{k-1}$ according to the system’s objective. Common performance metrics include energy consumption, material costs, availability of nodes, fidelity of sensor readings, application performance or common QoS Metrics. Note that these metrics may express both, beneficial and non-beneficial aspects corresponding to fidelity or costs as mentioned above.

These individual metrics are merged into a single fitness value for each scenario $S_i$. The method used in this contribution, is summing up $k$ metrics $m_j$ in the following form

$$f_{\text{fit}}(x, E) = \frac{1}{n} \cdot \sum_{i=0}^{n-1} \sum_{j=0}^{k-1} c_j(m_j(x, S_i))$$

(2)

where $m_j(x, S_j)$ denotes the outcome for a single metric regarding a scenario $S_j$. To make the metrics comparable regarding e.g. units, cost or fitness type metric, or simply order of magnitude, it is often useful to apply a conversion $c_j : \mathbb{R} \rightarrow \mathbb{R}$. The conversion $c_j$ might also apply a weighting of individual metrics. This may be a simple weighting $c_j(y) = w_j \cdot y$ or more complex operations. E.g. $c_j(y) = w_j \cdot \log(y)$ linearizes an exponential behavior of a metric. Other more complex weightings e.g. to emphasize certain intervals are conceivable. In any case $c_j$ determines the impact of the respective metric on the fitness value which is problem specific. Finally the fitness values for the individual scenarios are merged by calculating a mean. This might be enhanced by e.g. a weighting of the scenarios, the selection of the worst result etc.

$$E = \{S_0, \ldots, S_{n-1}\}$$

The returned result is a sequence of real values, each expressing the fitness value for the according individual of the population. The length of this sequence obviously depends on the number of individuals per population generated by the EA.

The first provides access from the specific implementations of EAs to the fitness adapter components on the basis of a generic interface. The second and third are engaged before running the simulation, while the latter two are producing the resulting fitness values by processing the output of the simulation runs, as depicted in figure 5. The configuration for the fitness adapter itself is defined by the fitness adapter configuration (FAC).

1) Interfacing: Since the EAs in use are existing implementations a generic interfacing between EA and fitness function needed to be determined. The interface defines a population as a sequence of vectors (individuals) consisting of a fixed number of real values. The dimension of these vectors is implied by the number of parameters contained in the set-up. The number of vectors is variable and depends on the number of individuals per population generated by the EA.

The returned result is a sequence of real values, each expressing the fitness value for the according individual of the population. The length of this sequence obviously depends on the number of individuals of the current population.

To enable a new type or implementation of an EA, one needs to implement a wrapper for it, that implements the according interface. The wrapper transforms the EA specific representation of individuals into the generic representation of set-up instances, which the ESAC toolkit uses to feed the fitness function. As the fitness functions evaluates this generic set-up instance, EAs are seamlessly exchangeable without need to change fitness function implementation.

2) Simulation Preparation: The simulation preparation module takes a population from the EA interfacing and generates executable simulation instances from these. Each of these is self contained and may be seen as workunit (WU). For
this preparation process a template for simulation instances is used which consists of all necessary executables, libraries and resources. The latter may for example contain sensor data which can be recorded by the nodes during simulation execution. Furthermore the simulation template provides a configuration file defining all needed parameter values except the ones provided by the set-up instances, i.e. the environment. Which parameters are set by the set-up instances is defined in the FAC.

Note that the evaluation of one set-up instance may result in multiple WUs if multiple scenarios were specified (compare for \text{V-A2}). This behavior, the affected parameters and their settings for the scenarios are configured in the FAC.

This module is actually reusable for different optimization tasks if they are based on OMNeT++. Only the simulation template and the FAC must be adapted accordingly—the effort to do this is comparable to preparing a single simulation execution outside the proposed optimization process.

3) Simulation Execution: The simulation execution component is a generic component, which processes the WUs provided by the simulation preparation component. As the executions of simulations are independent from each other, these WUs may be processed in parallel—anyhow a population must be completely evaluated before a new one can be generated. We provide two different modes, one which executes the simulations locally with multiple instances at a time—leveraging multi-core systems—and another based on the Berkeley Open Infrastructure for Networked Computing (BOINC, see [1]).

The BOINC architecture was designed to distribute tasks with extensive computational effort to a large set of computing nodes. These nodes are running a BOINC-client which may be registered to one specific BOINC-application (e.g. “seti@home” or “Climateprediction.net”) which is linked to a specific BOINC-server. The ESAC-toolkit can be see as an–until now–non-public BOINC-application. This server provides the necessary executables and according WUs. Clients download these at their initiative, evaluate the downloaded WUs and upload the results back to the server. The server monitors this process, checks uploaded results and provides statistics according to the gained progress.

As could have been expected, distribution based on BOINC induces some overhead and latencies due to the distribution via network, possible timeouts of the (polling) clients and the rasterization of WUs by populations. Hence it is especially applicable for runs with large populations, many scenarios and/or long running simulations.

Regardless of which execution mode was selected, the simulation execution component collects the log-files produced by the simulation execution and forwards them to the metrics processor for further treatment.

4) Metrics Processor: Log-files written by the simulation are supposed to contain the data relevant for the chosen metrics $m_j(x, S_j)$. Currently we have implemented three different parsers for log-files: one specifically developed for the first example below and two supporting the OMNeT++ specific vector and scalar file types.

ESAC requests the metric values through call-back functions, which must be implemented by the user since the metrics are application specific. Furthermore this component applies the metric conversions $c_j$. The metrics and conversions are declared in the FAC.

5) Fitness Processor: For each scenario a fitness value is calculated based on the processed metric values by applying the weighted sum. To merge the scenario specific fitness values to a single fitness value for the according set-up instance, the fitness processor calculates the mean (see equation [2]).

The fitness processor is entirely configured by the FAC and no further implementation is necessary. When the calculation has been finished for all individuals of the population the result vector is handed over to the EA interface.

VI. Visualization

Plotting recorded data is a standard procedure since it helps humans to quickly gain an overview of what is going on. Nevertheless preparing such plots may be a time consuming task e.g. due to the need to filter the data first or chose appropriate plotting configurations.

In the case of the proposed optimization process we are in the pleasant situation that the structure of the produced data is either always the same, e.g. for statistics output of ESAC or CMA-ES, or the structure and relevant parameters may be derived from the FAC. This enables us to provide the possibility to instantaneously plot the achieved results, even while the optimization process is still running. This saves time, reduces the chance of errors and provides a good overview on what happens at the optimization at all times.

The ESAC-toolkit provides two options for plotting: first a web interface and second a locally executed GUI-based application. While both are using the GNUplot tool [19], the web interface is meant to provide a quick overview with the most common plots at every time and everywhere—no option for extensive user interaction is provided. On the other hand the GUI-based application provides a large variety of plotting options comprising e.g. 2D or 3D plots, single or multi data-row plots and export options. All available data sources are recognized automatically and presented via drop-down lists. Hence it is very easy to gain a comprehensive overview with low effort.

Typical plots of interest are for example

- the fitness progression over generations,
- the fitness landscape for certain parameters or
- the parameter evolution and convergence.

The first helps to estimate if further optimization may be useful or if the gain on fitness stagnated over the last generations.

The second is very significant as it provides an idea about the shape of the fitness landscape and the influence of parameters. Note that the search space is typically more than two dimensional which prohibits to plot the complete landscape. Instead a projection of one dimension of the fitness landscape is plotted. Figure [6] depicts an example for such a projection.
### VII. Examples

The case study discussed in this section consists of two simple examples which might be conducted in an actual application of the ESAC toolkit in the following sequence:

**Example 1: Radio Interface**

We want to investigate the radio behavior, the nodes simply send out beacons to test for other nodes in range. If beacons are received these events are logged to a log-file. Furthermore traces of the nodes positions are recorded along with an according timestamp obtained from the GPS-devices (hence these timestamps are synchronized and therefore comparable between the nodes).

The first step to a meaningful simulation is an appropriate choice of the simulation model. We decided to use the IEEE 802.11 model of the INET framework first for simplicity reasons and second because it is a “standard” model. Nevertheless it should be mentioned that it is quite simplistic as the reception power is calculated based on the Frii’s equation and a purely radial radio propagation is assumed probably sub-optimal for the urban area.

To resemble the motion of the nodes we use the mobility

<table>
<thead>
<tr>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.2</td>
</tr>
<tr>
<td>0.3</td>
</tr>
<tr>
<td>0.4</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>0.6</td>
</tr>
<tr>
<td>0.7</td>
</tr>
<tr>
<td>0.8</td>
</tr>
<tr>
<td>0.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>150</td>
</tr>
<tr>
<td>200</td>
</tr>
<tr>
<td>250</td>
</tr>
</tbody>
</table>

The exemplary usecase is a WSN consisting of mobile nodes traveling through an urban area of approximately $\frac{1}{2}km^2$. The nodes are carried by pedestrians or cyclists which are restricted to the road-grid. Nodes therefore travel at moderate speeds and sample e.g. temperature and humidity data to capture the micro-climate of the urban area. The nodes are communicating via 802.11 radios operating in ad-hoc mode and route their data to two fixed data sinks located in the operations area.

To show up the different behavior and characteristics of GA and CMA-ES, we applied both of them to the first example. Since the GA performs somewhat more robust compared to CMA-ES which is supposed to be quicker, we chose to apply the GA first, narrow the search space and apply CMA-ES.

![Fig. 6. One of n parameters plotted against fitness. Each point represents an evaluated individual.](image)

![Fig. 7. Mean and standard deviation of a parameter that does not converge to a stable value during the optimization process.](image)
model presented in [16]. It provides on the one hand the possibility to replay the node movements as recorded in the real world experiment very precisely. On the other hand it allows to let nodes be move randomly on a simulated road grid. The first is what we use in this example, the latter what will be used in the next example.

**Fitness Function:** The most significant characteristic of a radio from a generic routing protocols point of view is if there is a connection between two nodes or not. Even if one can think of more sophisticated characteristic we leave it like that for this case study. The recorded log-files enable us to extract patterns of encounters between nodes by combining continuously received beacons. Beacons with a temporal distance less than a certain threshold and originating the same sender form a group. Each resulting group of received beacons represents one encounter between the sender $i$ and the receiver $j$ (where $i, j \in N$ and $N$ the set of nodes). We call the sequence of encounters with related temporal information between $i$ and $j$ the timeline $T_{i,j}(t)$. $T_{i,j}(t)$ may be understood as function that takes the values 1 and 0, expressing that communication from node $i$ to node $j$ is or is not possible. Note that $T_{i,j}(t)$ describes a unidirectional link only, and that the timeline $T_{j,i}(t)$ does not need to be identical. A complete encounter pattern for a network with $|N| = n$ nodes is represented by $n \cdot (n - 1)$ timelines.

Determining 'good' or 'bad' parameter values means comparing the according timelines $T_{i,j}^{\text{ref}}(t)$ from the reference execution—in our case recorded during the real world experiment—and $T_{i,j}^{\text{rep}}(t)$ from a replay execution based on the simulator (where $i, j \in N, i \neq j$). The mismatch between two timelines is exemplarily depicted in figure 8. The sum of mismatches $M$ between all according timelines of the reference and replay execution provides the metric on the quality of the match between reference and replay execution and may be written as:

$$M = \sum_{i \in N} \sum_{j \in N, i \neq j} \int_{t_{\text{start}}}^{t_{\text{end}}} |T_{i,j}^{\text{ref}}(t) - T_{i,j}^{\text{rep}}(t)| \, dt \quad (3)$$

This value of mismatch is the negative fitness value used for the optimization process and returned by the fitness function. It is calculated based on the measurements recorded during the simulation execution, i.e. the log-files written by a simulation run. Note that the described metric is the only one in this case, hence the merging of metrics using a weighted sum is trivial in this case.

**Set-up:** The parameters and parameter ranges used are shown in table 1. Note that one may guess the ranges quite freely, as could be seen in the case of the pathloss exponent, where a pathloss smaller than 2 is pretty much senseless when looking from the modeling perspective. Nevertheless one can

<table>
<thead>
<tr>
<th>Parameter</th>
<th>GA set-up</th>
<th>CMA-ES set-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitrate</td>
<td>fixed</td>
<td>fixed</td>
</tr>
<tr>
<td>Carrier Frequency</td>
<td>fixed</td>
<td>fixed</td>
</tr>
<tr>
<td>Transmission Power [mW]</td>
<td>1 to 100</td>
<td>20 to 140</td>
</tr>
<tr>
<td>Thermal Noise [dBm]</td>
<td>-80 to -150</td>
<td>-110 to -150</td>
</tr>
<tr>
<td>Sensitivity [dBm]</td>
<td>-40 to -100</td>
<td>-70 to -90</td>
</tr>
<tr>
<td>SNIR Threshold [dBm]</td>
<td>1 to 15</td>
<td>1 to 15</td>
</tr>
<tr>
<td>Pathloss Exponent</td>
<td>1 to 10</td>
<td>2 to 6</td>
</tr>
</tbody>
</table>

**TABLE I**

SET-UP INSTANCES FOR THE GA- AND CMA-ES RUN.
give it a try since the optimizer will point out which results are matching the real world reference data best.

Parameters like the carrier frequency or the used bitrate are predefined and must not be included in the set-up. Note that some parameters like e.g. the transmission power only seem to be predefined too. The power actually emitted by the device will typically vary between different products and possibly even between different charges of devices. Hence the values configured at the device may only be a guess but do need not be the 'truth'. Note that we nevertheless assume that the radios used in our set of nodes perform similar and we do not optimize the settings for each individual radio but for the type of radio.

Having a look at the available parameters, one can notice that the parameters are not independent and that there may be critical set-up instances, e.g. regarding the sensitivity of the radio and the thermal noise. If the thermal noise outperforms the sensitivity the radio will always be in the receive state.

**GA Run:** For the GA optimization run we defined a population size of 30 individuals. The larger the population the more robust the GA will behave regarding possibly noisy reference measurements, as e.g. real world data.

Figure 9 shows the fitness values found plotted against the actual parameter value. Note that the individuals or set-up instances are 5-dimensional, possibly resulting in five plots. We exemplarily picked two for presentation.

The results show that the pathloss coefficient is by far the most crucial parameter. Setting it below 2.5 leads to dramatically bad matches. This is due to the fact that below some threshold of the pathloss coefficient a node can communicate with every other node of the network leading to essentially one permanent encounter. This marks in practice the upper bound for the fitness function and may be calculated as

\[
\frac{(t_{\text{end}} - t_{\text{start}}) \cdot n \cdot (n - 1)}{\text{total length of timelines}} - \sum_{i \in N} \sum_{j \in N, i \neq j} \int_{t_{\text{start}}}^{t_{\text{end}}} |T_{i,j}(t)| \, dt
\]

With a total sum of about 4,100 seconds of encounters recorded in the reference data this leads to about 28,300 seconds which actually matches with the plot. Setting it greater than 3.5 leads to poor results where almost no encounters occur in the simulation, resulting a a fitness value converging against the sum of durations of encounters comprised in the reference data. All other parameters have less influence which is reflected by the uniformly distributed measurements without distinctive tendencies. Nevertheless it seems to be suitable to narrow the intervals of the parameters for the CMA-ES run as shown in table 1. Only the interval for the transmission power may possibly be chosen too rigid regarding the upper bound since there are a lot of good hits in this area. Hence we widen it for the CMA-ES run. For the remaining parameters the intervals are adapted accordingly.

The best individual found by the GA after approximately 1,000 simulation executions had a fitness of 2,260 seconds. Even more executions might give better results but to gain an overview this seems to be sufficient.

**CMA-ES Run:** As we pointed CMA-ES to the reasonable areas to search by scaling down the interval ranges of the parameters, CMA-ES follows the gradients of the fitness landscape quite well. Hence it spends more evaluations on the 'relevant' spots of the search space. This behavior is reflected by the very densely sampled regions in figure 10 for the pathloss coefficient and the transmitter power. On the other hand the very sparsely investigated areas reflect the issue that CMA-ES may relatively easy be mislead. These areas are not tested for good solutions thoroughly enough to make sure that they do not comprise good solutions.

Similar to the plots of the GA run, the relevant area is narrower in the case of the pathloss coefficient than regarding the transmitter power. This also holds for the other parameters and supports the observation, that the pathloss is the most significant parameter and can be fixated quite well. Nevertheless the transmitter power may also be determined in a range between 65 and 90 mW.

As mentioned in **III-B** CMA-ES provides some additional information regarding the status of the current optimization process and the reliability of the results. Figure 11 exemplarily shows the mean values and according standard deviations regarding the pathloss coefficient. In contrast to figure 7 CMA-ES is able to determine an appropriate value very quickly. After approximately 80 generations with 7 individuals each, i.e. after 560 evaluations, the standard deviation is near zero.
and the mean does not change significantly any more. Since the other parameters are less significant they can be fixated later, i.e. around generation 120.

The correlation between parameters may be derived as described in III-B but are also reflected in the distribution of individuals in the search space. Figure 12 plots positions of individuals in the search space regarding the thermal noise and the SNIR threshold parameter. One can recognize an ellipsoid similar to that depicted in figure 4 representing the major search area. Since this is adapted to the fitness landscape changing e.g. the SNIR threshold for an individual may be compensated by an according change regarding the thermal noise. Hence these parameters are correlated.

The finally best mismatch found by CMA-ES after 150 generations is approximately 2,080 seconds which is actually a bit better than the result obtained using the GA. Even applying both EAs as described here takes only about 2,000 simulation executions. This results in less than 6 hours of simulation in total for the discussed course of action at a midsize dual-core PC. Collecting real world data is in comparison to that the way more time consuming task.

Crosschecking: Since the achieved error values are quite high, we choose to take a short excursion: we crosschecked our approach by producing reference data in the clean environment of the simulator using the denoted radio model with some fixed set-up instance. Nevertheless some imprecisions remain, since the simulated GPS-device samples the position at 1 Hz. This leads to a slightly different node motion in the replay. Nevertheless the achieved mismatch was between 100 and 200 seconds, depending on the previously chosen reference set-up instance. As one may have expected the path loss exponent could be determined quite precisely with an error always below 5%. The other parameters are in contrast somewhat undetermined since the model allows to outweigh one by another.

One finding of this might be, that the optimizer and especially CMA-ES is not only able to search appropriate setup instances but also to bring some insights about possibly complex models.

B. Example 2: Routing Protocol

We have chosen the routing protocol OLSR as an example for an instance of a higher layer, depending on the radio as parameterized in example 1. An implementation of this protocol comes with the INET framework extended for MANet simulations [2]. Nodes send data packets periodically to one of two fixed sinks, which is randomly chosen for each packet. In this application we have two important modules which are depending on random values, namely the application choosing a sink and the mobility module generating a way through the road grid. To make regarding decisions stochastically independent and therefore to mitigate adverse effects we supply each of these modules with its own random number generator. Furthermore we use three scenarios with different seeds as proposed in V-A2.

Fitness Function: Two common network performance metrics are the delivery rate and the overhead caused by the routing protocol and the underlying layers. For a set-up instance \( x \) we measure the delivery rate \( D(x) \) as the ratio of the number of delivered application-data packets to the number of sent application-data packets in the whole network. The overhead, measured as the quotient of all bytes sent through the network \( B_{\text{mac}}(x) \) to the delivered application-data bytes \( B_{\text{dlv}}(x) \), is \( O(x) = \frac{B_{\text{mac}}(x)}{B_{\text{dlv}}(x)} \). The value \( B_{\text{mac}}(x) \) is counted at the gate between MAC layer and physical layer and summed up for the whole network, such that the overhead increases with the number of management packets as well as with a larger number of hops. The fitness function combines these two performance metrics in a sum with equal weights.

While \( D(x) \) is bounded between 0 and 1, \( O(x) \) has no upper bound and is always greater than 1. This means that the overhead metric would always be the dominating value compared to the delivery rate.

Therefore we apply a conversion operation \( c_O(o) \) to the overhead metric, that linearly maps overhead values \( o \) into a range between 0 and 1 if \( o_l \leq o \leq o_u \). The lower limit \( o_l \) is set to 1, as this is the ideal case of no overhead and the upper limit \( o_u \) is set to a relatively high value of 30. Values less than \( o_l = 1 \) can not occur and values greater than \( o_u \) will be mapped to 1.
Because the Optimizer tries to minimize the fitness value, we define \( c_D(d) = 1 - d \), so that the ratio of non delivered packets is summed up with the overhead. Finally the fitness calculation for a single scenario can be formulated as

\[
f(x) = c_D(D(x)) + c_O(O(x))
\]

where \( 0 \leq f(x) \leq 2 \). The resulting fitness values of the three scenarios are merged by applying a mean.

**Set-up:** The implementation used in the INET framework comes with only a few parameters and we reduced the set-up to two of them to keep the example more handy:

1) **Hello interval**, defining the emission frequency of messages used for link sensing and neighborhood detection.
2) **Topology change interval** (TC interval), that clocks the propagation of information about possible connections.

In contrast to example 1 we applied only the GA since it provides a widish distribution of samples leading to a more uniform coverage over the search space. Together with the only two dimensions this allows us to plot a 3-dimensional fitness landscape.

**Results:** Figure 13 shows the according results. Areas of good fitness values are indicated by dark, areas of worse fitness values by brighter shades of gray. The surface plotted represents the best achievable fitness value, i.e. the lower bound. It is observable that a good value for the hello interval lies between 5 and 6 seconds while the TC interval should be at least greater than 4 seconds. The best set-up instance found lies at approximately 6 ms for the hello interval and 5.8 ms for the TC interval.

Figure 14 shows in the upper part the projection of the fitness regarding the three individual scenarios over values of the hello interval. Note that the scenarios only differ in the selection of seeds and result in significantly different absolute fitness values. Nevertheless the characteristics are similar, especially regarding the separation into two ‘layers’ of either bad or good results but only few inbetween. Merging the scenarios leads to the lower plot of figure 14 which represents the fitness landscape as observed by the EA. The mentioned minimum at approximately 6 ms is clearly visible. Furthermore one can recognize the four ‘layers’ induced by merging the individual scenarios with their two ‘layer’ characteristic.

**VIII. RELATED WORK**

Simulation-based optimization is a well proven method in the industrial area. E.g. in [20] Schneider et. al. introduced a **modular approach for simulation-based optimization of microelectromechanical systems**, that is based on a generic framework that couples implementations of optimizers, simulation techniques, visualization and analysis modules. This framework is conceptually similar to the framework we introduced. They also took evolutionary approaches into account and provided a method for the distributed execution of all components.

Barth et. al. presented in [6] a software architecture for distributed solution of simulation-based optimization problems in engineering applications. They also modularized and coupled the typical components of such optimization processes via a framework, but more focused on the distributed execution of simulations.

In the field of (wireless) sensor networks, simulation-based optimization seems to be less common than in the industrial area. Simon et. al. presented an approach, focussing on the optimization of communication protocols for large-scale WSNs in [21]. In this context they developed their own simulator and an optimization framework (including an own optimization algorithm), which did not decouple the algorithm from the rest. This is an essential difference to our contribution, as we utilize ready-made algorithms and simulation techniques, in order to increase flexibility, reusability and reduce programming effort and error sources.

Evolutionary approaches for configuration and planning are nevertheless relatively often used in the area of WSNs for very specific tasks. E.g. Sajid et. al. presented in [14] an approach to enhance a hierarchical WSN’s clustering with regard to the energy consumption and lifetime of the network. Their results are promising to other well known routing approaches. Similar to that Ferentinos et. al. propose in [8] to additionally involve
application specific parameters and optimize the placement of nodes, too. Likewise Zhao et al. [24] and Jourdan et al. [15] deal with the placement of nodes, solving the best-coverage problem based on genetic algorithms. The latter is using a multi objective genetic algorithm to provide pareto optimal solutions. All of these contributions are using GAs exclusively and therefore they cannot take advantage from the combination of different optimization algorithms. As mentioned they focus on specific tasks—like clustering—or try to optimize with regard to specific applications only, like Ferentinos et al. for an agricultural scenario.

In a similar way Gao et al. propose an approach to optimize the placement of sensor nodes for structural health monitoring with regard to a minimal missdetection probability [9], which is actually based on CMA-ES. Nevertheless most of the according contributions are based on genetic algorithms and focus on issues directly or indirectly related to the radio interface. Outside the field of sensor networks a reasonable amount of work has been based on GAs to optimize radio communication scenarios, e.g. to place base stations [10], [18].

IX. CONCLUSION

The presented approach obviously does not come at no costs since providing reference data, the preparation of a fitness function, running the simulations, the interpretation of results etc. might be seen as an additional overhead. Nevertheless we advocate that this effort is well invested since it contributes to the trustworthiness and reliability of WSN simulations. It is beyond all question that expert knowledge is very valuable but defining e.g. a radio set-up instance for simulation without measuring the real world will be an estimation only—even if based on expert knowledge. Additionally it is not uncommon that default settings for simulation models are somewhat unrealistic and they are in particular not adapted to the specific application. Hence they should be used for toy problems only. The trustworthiness of simulation results will grow more and more important as WSNs are actually getting deployed for real applications. ESAC provides a way to methodically determine parameter values related to measurements and formally defined goals to achieve.

Besides the pure trustworthiness of produced results we observed other unintended sideeffects: On the one hand the definition of a fitness function demands the developer to really thoroughly think about and define goals and thresholds for the WSN which is going to be deployed. On the other hand implementations of models, protocols and applications are getting tested quite thoroughly even under uncommon (but still possible) conditions which few would have thought about. Both issues turned out to be quite beneficial for the development process—even apart from the pure task of configuration.

The original intention of the discussed approach was to find reasonable parameter values for simulation models, protocols and applications. Nevertheless these near optimal set-up instances constitute the best results achievable regarding the given fitness function. Hence they provide something like a 'normalization' and allow to compare simulation models against each other or rate models according to some reference data reflecting real world situations (see [17]). We can possibly answer questions like: Which radio model resembles the real world according to specific needs best, or which is the right routing protocol for a specific application?

REFERENCES

Middleware Support for Dynamic Reconfiguration in Sensor Networks

Paul Grace, Danny Hughes, Barry Porter, Paul Alcock, Geoff Coulson, Gordon Blair
Computing Department, Lancaster University, UK
Email: {gracep, danny, barry.porter, {p.alcock@lancaster.ac.uk}, geoff, gordon}@comp.lancs.ac.uk

Abstract—The operational environments of sensor networks will alter over time, often due to hazardous conditions or fluctuating resource availability; however this important characteristic of sensor networks has yet to be fully addressed by current middleware solutions. In this article, we present a reflective middleware solution for co-ordinated dynamic reconfiguration of middleware behaviour across nodes in a sensor network. We evaluate this approach in a real-world case study; firstly, we demonstrate how dynamic reconfiguration optimises the performance and resource consumption of a sensor application, and secondly we illustrate that the costs of reconfiguration are not prohibitive in this domain.

I. INTRODUCTION

Wireless sensor networks are now employed in a wide range of application domains including, for example, environmental monitoring and disaster management. The role of sensor middleware is to support the application developer, and shield her from: i) the complexity of developing applications on heterogeneous low-level hardware such as MICA Motes, Gumstix, or bespoke sensor technology, and ii) routing messages across heterogeneous ad-hoc networks e.g. Bluetooth, 802.11b, Zigbee, and others. Initial sensor middleware solutions have concentrated on common communication abstractions (e.g. event subscription [1], database-style [2], or tuple-spaces [3]) over a variety of ad-hoc message routing strategies [4]. Some proposals have additionally considered the management of resources within sensor networks e.g. power consumption [5]. However, the dynamic change of a sensor network’s operational environment and general context has yet to be considered. For example, sensor network middleware used in disaster scenarios will typically need to react to increasingly hazardous conditions e.g. flooding in a monitored location may cause the network to perform poorly or fail. Therefore, we believe that sensor middleware should be inherently adaptive.

In this paper we examine two key aspects of our Gridkit middleware [6], which support dynamic reconfiguration in sensor networks:

- Pluggable routing protocols, and overlay networks. In Gridkit, sensor networks are created as virtual network topologies (i.e. overlay networks), which can be selected to support different application requirements. For example, an overlay that conserves power may be most appropriate for one application type, whereas an overlay that focuses on robustness might be better in a different setting. Different overlays optimise for different characteristics, primarily by differing in terms of their topology and in how messages are routed between nodes.

- Co-ordinated Dynamic Reconfiguration. To support reconfiguration of overlay behaviour, co-ordinated adaptation of the per-host middleware across all nodes in the sensor network is required—e.g. the routing protocol on every node must be updated. Gridkit introduces the concept of distributed frameworks for this purpose; these consist of a set of sensor nodes and a set of distributed reflective meta-protocols that allow i) the inspection of the current network-wide middleware configuration, and ii) the coordinated reconfiguration of software elements across nodes.

To evaluate our middleware we use Gridkit to deploy a real-world sensor application, involving the management of flooding in a river valley in the North-West of England. We concentrate in particular on how our middleware can perform system wide adaptations of the overlay networks to react to changing conditions such as increasing water flow, and sensor failure caused by flooding. We demonstrate that these reconfigurations are beneficial to the operation of the sensor network compared to static deployments, and they do not come at a prohibitive cost in this type of application.

The results in this paper build upon previous work in [7] where we reported on our general approach to supporting distributed reconfiguration in sensor networks. This paper significantly extends this work, with a quantitative evaluation of our approach on the GumStix embedded computing platform and in a real world WSN-based flood predictions scenario. We also in [8] reported on how the GridStix hardware and software platform has evolved over time, however this paper focuses more upon a detailed evaluation of the 'production' GridStix 1.0 platform as was deployed at our River Ribble site from 2005 - 2007.

II. GRIDKIT SENSOR MIDDLEWARE

Gridkit [6] is a generalized middleware framework that can be specialized to operate in diverse application types (e.g. Grid computing, pervasive computing, and mobile computing). Here, we examine the specialization to the domain of sensor middleware. Fundamentally, Gridkit is constructed as a set of components developed using the OpenCOM v2 [9] component model. This employs a minimal runtime that supports the loading and binding of lightweight software components at run-time.
Figure 1 illustrates the key elements of Gridkit’s wireless sensor network profile. Applications use the interaction framework that contains a customizable event service. A node can be just a publisher, just a subscriber, or, optionally, it can act as a broker in the event service. This is then layered above a core distributed framework known as the overlays framework. This hosts, in a set of distributed overlay framework instances, a set of per-overlay plug-in components, each of which embodies i) a control element that cooperates with its peers on other hosts to build and maintain some virtual network topology, and ii) a forwarding element that routes messages over its virtual topology. Different overlays can be employed here depending on the conditions, e.g. a distributed spanning tree overlay, or a proximity aware overlay. Each overlay can be dynamically reconfigured to react to changing conditions (by changing the control and/or forwarding component). For example, figure 1 shows an example reconfiguration of the distributed spanning tree overlay; we can reconfigure the topology of the overlay from a shortest-path tree to a fewest hop tree by replacing the control component. This process is performed across all nodes who are members of the overlay using the procedure described in the following section.

III. DYNAMIC RECONFIGURATION

There are two important dimensions in the dynamic reconfiguration of sensor networks: local and distributed. Local adaptation is the dynamic reconfiguration of software elements on an individual node; whereas distributed adaptation is the adaptation of the behaviour across a sensor network. Therefore, a distributed adaptation consists of a series of local adaptations. In Gridkit, dynamic reconfiguration is based around software architecture elements known as local and distributed component frameworks.

A. Local Component Frameworks

The local component framework model (illustrated in figure 2) is based on the concept of composite components as proposed in the OpenORB project [10]. Each framework has a reflective meta-interface (ICFMetaInterface) that enables inspection and dynamic adaptation of the local ‘architecture’ of the composite component in terms of its local components and connections. Additionally, the integrity of each framework is maintained in the face of dynamic change, using developer specified architectural rules plugged into the component framework (through the IAccept interface).

The second aspect of the local component framework model is the use of the configurator pattern [11] as illustrated in figure 2. A configurator is assigned to each framework instance, and acts as a unit of autonomy for making decisions about when and how to change the framework. Each configurator maintains a set of local policies for its framework. It is connected with the Gridkit context engine [6] to receive relevant environmental events; and communicates with its host framework through the meta-interface. Gridkit policies use the Event-Condition-Action pattern. When an event is detected, it triggers the corresponding action, which is a reconfiguration script of component inserts, deletes, disconnects, connects, or replaces.

B. Distributed Component Frameworks

A distributed component framework (illustrated in figure 3) is a set of local component framework instances of the same type located across a set of co-ordinated devices, typically providing co-ordinated middleware functionality. The design of the distributed framework model follows the same basic themes as for local frameworks - i.e. the use of reflection to support inspection and adaptation of software, and configurators to enforce autonomic actions. The distributed component framework model must be inherently more flexible than the local model, as there are many more constraints that disallow a single fixed model being utilised. We now discuss the important elements of distributed frameworks in turn.

- Meta-Object Protocol & Reification Strategies. Each distributed framework maintains a basic meta-object protocol (MOP) that reifies the information about the global contents of the framework; this can be in terms of node
members, and also the component configurations on each host. The MOP provides operations (through IDistributedMetaArchitecture in figure 3) for the insertion and deletion of local framework elements into/from a given distributed framework, and the inspection of information about the set of individual participants.

- **Group-based membership support for the MOP.** We use a lightweight group membership service as the base mechanism for distributing meta-data and reconfiguration events; this data then builds the view of the system wide architecture. The protocol is customisable: typically different group membership overlays will suit different sensor models—e.g. a sensor network with high node mobility will require a different group membership overlay from a more static network topology. So far, we have used the scalable membership protocol SCAMP [12] to maintain meta-data between members of a distributed framework.

- **Configurators.** Distributed configurators (as seen in figure 3) follow the same pattern as for local frameworks. They receive events about changing environmental conditions, select policies and then perform distributed reconfigurations using the MOP. However, individual frameworks may have more than one configurator (e.g. there could be one on every node). Therefore, consensus protocols are used to ensure that all members of the framework agree on the action to perform. Our evaluation has so far focused on single configurators; however, we are also investigating the introduction of selectable and replaceable consensus algorithms into our distributed frameworks.

- **Quiescence.** For safe dynamic reconfiguration it is important to ensure that updates complete atomically and do not impact the integrity of the network. There are two parts to this: i) making the framework safe to adapt, i.e. placing it in a quiescent state, and ii) ensuring that the reconfiguration is complete and correct. In our current implementation, local framework instances maintain a readers/writers lock to place it into a quiescent state: any standard interface call or meta-inspect is a reader, any meta-write is a writer. Hence, no reconfiguration can take place locally while a thread is executing in the framework. Currently our distributed frameworks use this capability; each local framework is placed into a quiescent state through a command propagated via the meta-group service. Once locally quiescent a notification is returned to the configurator. Upon the condition that all members are in a quiescent state then the reconfiguration continues. After reconfiguration, like with local frameworks (IAccept plug-in), the update can be checked through inspection of the meta-data to validate the integrity of component updates across multiple nodes. The disadvantage of this approach is that it may be too resource intensive, and may not scale suitably for large numbers of hosts. Therefore, we are investigating replaceable decentralised strategies for safely updating components.

**IV. CASE-STUDY BASED EVALUATION**

**A. Background**

We now discuss the use of the Gridkit sensor middleware in an implemented real-world scenario: wireless sensor network-based real-time flood forecasting in a river valley in the north west of England. This scenario is described in additional detail in [13].

In this scenario, a wireless sensor network (WSN) comprising of 20 nodes has been deployed to monitor depth and flow conditions along a 2.5KM stretch of river in the Yorkshire Dales in North-West England. The system monitors Fig. 3. Distributed Frameworks (per host components)
water depth using pressure sensors and flow-rate using a combination of image-based flow measurement and ultrasound flow measurement. Sensor data is collected in real-time at one or more designated ‘root’ nodes and forwarded from there via GPRS to a prediction model that runs on a remote computational cluster. Each sensor node (known as ‘GridStix’) comprises a 400MHz XScale CPU, 64MB of RAM, 16MB of flash memory, and Bluetooth and WiFi networking hardware (the root nodes are also equipped with GPRS). Each GridStix is powered by 4 watt solar array and a 12V 10Ah battery. They run Linux 2.6, version 1.4 of the JamVM Java virtual machine and the Gridkit version 1.5 WSN profile (figure 1). The following section now discusses the role of reconfiguration in this scenario.

In the scenario we used Gridkit with the distributed spanning tree overlay plug-in, which is used to disseminate sensor data between a large number of sensor nodes and a small number of root nodes. This plug-in is configured to operate in two modes; fewest hop and shortest path:

- **Shortest Path (SP)** spanning trees are optimised to maintain a minimum distance in edge weights from each node to the distinguished ‘root’ node; in our case edge weights are derived from the power consumption of each pairwise network link; SP trees tend to consume less power than FH trees, but offer poorer performance.
- **Fewest Hop (FH)** spanning trees are optimised to maintain a minimum of hops between each node and the root; FH trees minimise the data loss that occurs due to node failure, but are sub-optimal with respect to power consumption.

The different properties of these overlays make them more suitable for different environmental conditions. During quiescent conditions, when the criticality of sensor data is low, the system is configured to use a shortest path spanning tree. Conversely, during flooding conditions, when the criticality of sensor data and risk of node-failure is high, the system reconfigures to use a fewest hop spanning tree which is more resilient and offers better performance.

### B. Evaluating the Role of Reconfiguration

The benefits of reconfiguration have been evaluated primarily through simulation of the Gridkit middleware in operation in an example sensor deployment. The GridStix simulator models the low-level properties of each node (available CPU, available Battery, solar panel power production) and each pairwise network link (round-trip-time, power-consumption, bandwidth, delay, jitter, loss). This low-level data has been measured empirically on the real-world system, which makes the simulator highly accurate for this scenario. The visualisation sub-system of the simulator is shown in Figure 4 illustrating FH (left) and SP (right) overlay configurations.

The simulation was configured as follows: The simulation period is 24 hours (midnight-to-midnight). Each node enters the simulated period with a battery at 50% charge. Flood conditions begin at 12PM and last until 6PM (the approximate mean duration of a flood event at the site). Dawn occurs at 8AM from which time solar power production is set to WINTER_SUN, when flood conditions begins at 12PM, solar power production is set to HEAVY_CLOUD, finally night falls at 8PM (approximately mimicking late winter conditions, when flooding is most prevalent). All nodes were programmed to wake for one minute in every hour. During quiescent conditions nodes transmit sensor readings at a rate of one per minute. During flooding condition, nodes transmit sensor readings at a rate of one per second. This is commensurate with the increased requirements of performing real-time flood modelling.

Throughout the simulated period, the system’s performance has been evaluated in the context of three key metrics:

1) **Performance**: We measure this in terms of the latency with which messages can be relayed from each sensor node to the root node.
2) **Resilience**: The resilience of the network is a function of the extent to which the failure of a given node reduces the overall connectedness of the network. We measure this as the number of viable routes between each node and the root.
3) **Power Consumption**: Although the GridStix are equipped with solar panels, power consumption is still an extremely important factor. We infer this from the per-node battery power consumed throughout the test.

In all cases we measure and plot each of these metrics averaged for all nodes in the network, at intervals of one minute throughout the duration of our trace. Figure 5 shows the battery life of each node in the system using an SP configuration, an FH configuration and an adaptive configuration, wherein when flooding is detected (at noon), the middleware reconfigured from using a low-power SP tree to a high-performance FH tree.

Figure 5 shows that SP trees ensure that nodes maintain the highest possible battery life throughout the test. Conversely, FH trees result in the greatest battery power consumption.
(though at the expense of performance and resilience as shown below). As one might expect, where the system reconfigures at flood time from an SP to a FH configuration (‘adapt’, shown in green), battery power is maximised during quiescent conditions (i.e. approximating SP), while increasing during flooding conditions (i.e. approximating FH) - though at the same time providing better performance and resilience, as shown in figure 6 and 7 respectively. Finally, the graph illustrates that the power consumed by reconfiguration, both in the transmission of reconfiguration messages and in CPU-usage is acceptably low. In fact, it is too small to be noticeable in figure 5.

Figure 6 shows the mean reporting latency throughout the test period. As one would expect, aside from normal jitter, FH and SP configurations remain relatively constant throughout the test. FH configurations demonstrate a mean reporting latency of 11ms, while SP offers a reporting latency of 28ms (though consuming significantly less power, as shown in figure 5 above). Finally, where the system reconfigures from an SP configuration during quiescent conditions to an FH configuration during flood conditions (‘adapt’, shown in green), performance is correspondingly low during quiescent conditions, but high during flood conditions (though at the expense of power).

Figure 7 illustrates the resilience of the system to node failure, based upon the mean number of routes that are affected by node failure. As FH trees have a typically lower node degree they tend to be more resilient to node failure, while SP trees have a typically higher degree and are therefore significantly more vulnerable to node failure. As one would expect, during quiescent periods, where nodes reconfigure between overlays during flooding, system resilience matches SP during quiescent periods and FH during flooding periods (i.e. it is more resilient, but at the expense of power).

In summary, testing on a real-world system illustrates the benefits of overlay reconfiguration. By configuring to a low-power overlay during normal conditions and a high performing and resilient overlay during flood conditions, battery life is extended, while maintaining system functionality during critical conditions. We also illustrate that the cost of reconfiguration in terms of additional power consumed, and reconfiguration time does not significantly affect the power costs and performance.
of the sensor network.

V. RELATED WORK

There are a number of existing sensor middleware, some of which were described in the introduction. These are important solutions in identifying the key characteristics required by sensor middleware, namely easy to use abstractions, suitable routing strategies, and resource management policies. However, none of these pieces of work considers dynamic reconfiguration and customisability to the same degree as Gridkit.

Outside the domain of sensor networks there are a set of technologies related to distributed reconfiguration. Kramer and Magee [14] describe algorithms for distributed reconfiguration that inspired our approach. k-Components [15] offers a decentralised agent-based approach; i.e. each node makes local decisions about adaptation, until a global consensus is reached. Gridkit supports both centralised and decentralised approaches, and in that manner can be tailored to resource-constrained domains such as sensor networks. NecoMan [16] offers an alternative approach to dynamic reconfiguration. It supports safe, co-ordinated updates of distributed services, typically related to network protocols. It has not yet been applied to multiple reconfiguration domains to illustrate its flexibility; however, it presents many interesting ideas that could be applied within our frameworks. Ensemble [17] is a micro-protocol stack framework that is able to adapt its configuration dynamically; however, the reconfiguration mechanism is closely coupled to the micro-protocol implementation, and isn’t as re-usable as Gridkit.

VI. CONCLUSIONS

In this paper, we have proposed the introduction of dynamic reconfiguration into sensor middleware to ensure that the behaviour of the sensor network is optimised in the face of fluctuating conditions. We have demonstrated how the flexible Gridkit middleware platform can be tailored specifically to the sensor domain. In addition, we have introduced the concept of reflective distributed frameworks to manage the reconfiguration of software components across the sensor network. The approach was evaluated in a real-world sensor application, illustrating the benefits of reconfiguration, which come with acceptable costs.

There are a number of interesting future areas of research inspired by this work. Firstly, the creation of higher-level declarative languages that can be used by both middleware and application developers to describe reconfiguration actions on the sensor network, and also deal with potential conflicts that may arise from multiple policies. Secondly, the introduction of security measures to the distributed framework to ensure only authentic nodes can join a framework, and only members of the framework can make reconfigurations.

REFERENCES


A Peer-to-Peer Framework for Globally-Available Sensor Networks and its Application in Building Management

Ioannis Chatzigiannakis, Christos Koninis, Georgios Mylonas
Department of Comp. Eng. & Informatics
University of Patras, Greece
Research Academic Computer Technology Institute, Patras, Greece
Email: {ichatz, koninis, mylonasg}@cti.gr

Ugo Colesanti, Andrea Vitaletti
Dipartimento di Informatica e Sistemistica
SAPIENZA Università di Roma
Rome, Italy
Email: {colesanti, vitale}@dis.uniroma1.it

Abstract—In this work we present the architecture and implementation of WebDust, a software platform for managing multiple, heterogeneous (both in terms of software and hardware), geographically disparate sensor networks. We describe in detail the main concepts behind its design, and basic aspects of its implementation, including the services provided to end-users and developers. WebDust uses a peer-to-peer substrate, based on JXTA, in order to unify multiple sensor networks installed in various geographic areas. We aim at providing a software framework that will allow developers to deal with the new and critical aspects that networks of sensors and tiny devices bring into global computing, and to provide a coherent set of high level services, design rules and technical recommendations, in order to be able to develop the envisioned applications of global sensor networks. Furthermore, we give an overview of a deployed distributed testbed, consisting of a total 56 nodes and describing in more detail two specific testbed sites and the integration of the related software and hardware technologies used for its operation with our platform. Finally, we describe the design and implementation of an interface option provided to end-users, based on the popular Google Earth application.

I. INTRODUCTION

The last few years we have witnessed the emergence of a growing trend towards the realization of a stronger connection between the natural and the digital worlds. This is a two-way association, with the digital domain on the one hand invisibly becoming a part of our everyday lives and interacting with human activity, and on the other hand we pursue monitoring in greater detail and scale natural processes and activities with the aid of the digital domain. Regarding these two general trends, lots of names and approaches have been proposed and adapted by the research community, e.g., ambient intelligence, pervasive computing, etc., while also given various contexts and interpretations.

Wireless sensor networks (WSN) have been proposed to deal with these two directions, especially the latter one. Over the years the initial definition of a WSN has loosen significantly, and requirements such as low-power, small-size, ad-hoc, etc., usually attributed to these networks are rapidly becoming somewhat “optional”, also based on realistic application requirements. As a result, the discussion is shifting more towards “distributed systems that collectively interact with the physical world”. Furthermore, whereas in the past we had isolated “islands” of sensor networks, we are now actively seeking sensor network interconnection. The WISEBED [1] research project is an example of this new approach.

In that respect, recent research in WSN strongly relates to Internet-scale sensing applications. Unlike the applications that were proposed or implemented in the previous years, interfacing between/with different networks and applications is gaining in importance. Such applications are related to what is referred to as “the Internet of things”. To realize such a vision, software for WSN applications must adapt to the new requirements posed. The seeming integration of the Internet and sensor domains and its benefits can be better described e.g., by the concept of a Sensor-oogle or a Google Earth Sense. One can imagine a version of Google offering search capabilities for sensing services or a Google Earth with live sensing data coverage per square mile.

Since there is already a massive amount of work on lower sensor network levels, two challenges emerge: a) the creation of systems that integrate, at least to a certain extent, the research results so far, in a way that allows the development of applications that use these results in an easy and efficient way, and b) since we are gradually using new sensor network models with multiple sensor networks each consisting of relatively few nodes and not single massive WSNs with thousands of nodes, we need to research the way these systems will be interconnected and how can we build applications over them. Some of the fundamental problems rising are:

- HW and SW heterogeneity,
- intermittent connectivity,
- application scaling capability,
- simplification of the development and deployment cycle.

We present here the basic concepts behind the design and implementation of WebDust, a peer-to-peer platform for organizing, monitoring and controlling wireless sensor networks. Our software architecture provides a range of services that allow to create customized applications with minimum
implementation effort that are easy to administrate. In this work we describe in detail its open architecture, the design decisions, and discuss various important aspects of the system’s implementation. WebDust aims to allow heterogeneous components to operate in the same sensor network, and also give the ability to manage and control multiple such networks, by means of defining web-based mechanisms to visualize and manage network state, user-defined queries’ results, and a means to inject queries in the network.

Our system is closely related to the AEOLUS research project [2], which investigates the principles and methods for building an overlay computer, consisting of computing entities connected to the Internet, providing a specific set of high-level integrated services. WebDust provides extensions for wireless sensor networks to this global computer by defining and implementing a range of services for discovering, querying and providing sensor network data and other related resources.

Regarding the structure of this work, in section II we outline the related work in the field, along with our contribution. In section III, we give an overview of the overall architecture of the system, while in the following section we give a more detailed view of the components comprising our system. Also, in section VI we describe a graphical user interface to the system based on Google Earth. In section VIII we give further details regarding the implementation and the operation of an experimental deployment, that was used for interconnecting discrete sensor network testbeds. We conclude in section IX and outline our future work.

II. RELATED WORK

There is an existing large body of work on WSN related software environments proposed or implemented the last few years. [3] and [4] are sources providing references to a number of related solutions so far. Some of these approaches are quite similar to our own, regarding the multitude of manageable sensor networks and large-scale system size. However, not all of the software solutions presented thus far deal with exactly the same issues, i.e., their respective application scenarios and target domains differ, or they choose a different architecture to solve the same problems.

The majority of software available follows an N-tier architecture (e.g. [5], [6]), while other, more recent, approaches extend this concept by using a peer-to-peer logic, to variable degrees. IrisNet [7] was one of the first proposals for dealing with large-scale sensor networks. CarTel [8] emphasizes more on coping with mobility at high speeds, while MetroSense [9] focuses on node intercommunication techniques, and Skylark [10] aims to integrate sensor networking in the mobile phone world. P2PBridge [11] and Agimone [12] focus more on the concept of inter-WSN communication and interoperability using software agents. Hourglass [13] is another peer-to-peer environment, which has a number of goals that are similar to our own, but targets a specific application domain.

A system with rather similar goals but different architectural approach, is SenseWeb[14]. It is used as a platform for publishing sensor network data from research teams that wish to make their data publicly available. It uses web services in order to provide an API for developers to connect to a central “datahub”, which can be used also to store data apart from advertising it. It uses a rather centralized approach, in contrast to our peer-to-peer notion of making sensor data available. Also, SensorMap [15] is a portal site that serves as a user interface, based on the Virtual Earth portal by Microsoft, a concept similar to our own user interface, but we also use 3D representations of buildings and other entities as to offer a different user experience, e.g., for offering simultaneous view of sensor readings produced in different levels inside buildings.

The OGC Sensor Web Development [16] initiative defines a set of standards and web services regarding the interconnection of sensor networks, while also proposing a general architecture. At this stage, we think that there is certain potential in this approach, but the proposed architecture is a bit too general, covering lots of different classes of devices, and also rather complex to integrate it into our own, although we intend to look further into such issues in our future work. We use similar concepts in our work, i.e., to advertise sensors and readings (like in SensorML), or register nodes and networks in a “directory”, but in a more simplified (and less descriptive) way.

There is also another growing trend in using Web Services or protocols like 6LoWPAN, in order to connect sensor networks with the Internet and expose their functionality in a more transparent way than previous approaches. Tiny Web Services [17] is an example of the first approach, while the software provided by Arch Rock [18] is based on a 6LoWPAN architecture. Our system provides a more service-oriented approach to interfacing with the rest of the world.

III. ARCHITECTURAL OVERVIEW OF THE SYSTEM

We consider global networks as peer-to-peer systems where human users, intelligent agents, and powerful computers interact with wireless sensor networks, ambient intelligence environments and smart spaces. In particular, we distinguish the global network into three sub-domains:

- The peer-to-peer network, where peers are applications executing on a powerful computer device that has the ability to communicate with other peers via the Internet or other global networks (i.e. entities of the overlay computer).
- The Nano-Peers (i.e., the sensor devices), that form wireless sensor networks and communicate via the wireless medium (or optical).
- The Gateway Peers, that consist of nodes that have access to the wireless sensor network and allow interaction between the actual peer-to-peer peers and the nano-peers.

The distributed characteristics of our system are achieved via a series of functions and mechanisms (services) which are triggered by the system as a response to various kinds of events, processes, actions, applications that take place on it. We define a service as a unit of work done by a service provider (i.e., sensor device, gateway, peer) to achieve desired
end results for a service consumer (i.e., gateway, another sensor device or a peer). Both provider and consumer are roles played by software agents. The results of such a service are usually changes in the state of the software agent of the consumer, but can also be a change of state for the provider or for both.

Rather than following a multi-tiered architecture, in which a number of tiers are operating in a specific hierarchical way and specific interfaces are used for communication across each layer, we chose to follow a less tightly coupled architecture in order to offer more flexibility to the system. Interaction among services is performed over all the levels of the system, in order to exchange necessary information for the successful accomplishment of each of them. Software agents (services) running on a peer are considered as independent modules that may interact with other services on the same peer and/or software agents executed by nearby peers.

The services related to monitoring and controlling the system’s performance interact with almost all other computational, communication, distributed, and other performance active mechanisms in order to receive necessary information for the correct evaluation and measurement of system’s performance. Services for failures handling interact also with almost all other active computational, communication, distributed, and other performance services/processes in order to maintain the system’s fault-tolerance. In order to achieve this, interfacing of the software agents is achieved via messages that are descriptive rather than instructive. We trade-off efficiency with extensibility by allowing the software agents to use a relatively rich vocabulary while interacting with each other.

Furthermore, due to the heterogeneity of the devices but also to the very nature of such global sensing systems, each peer may operate with a different set of software agents, i.e. provide a subset of the available services, and may provide different versions of a particular service, i.e. provide different quality and functionality. In this sense, we point out that:

- Each wireless device may operate different set of software agents, i.e., provide a subset of services.
- Each wireless device may operate different versions of a particular service.

The key point of our architecture are Gateway peers that appear on the global network as participants that have certain sensing and monitoring capabilities. The existence of the wireless sensor network is abstracted from the top layers and similarly the overlay computer is totally unknown to the nano-peers. In this way we categorize services in two levels:

- **Higher level services**, which are made available to the overlay computer by the gateway peers.
- **Lower level services**, which are used by the nano-peers (within the wireless networks), that are transparent to the overlay computer (i.e. “internal services”) and only available to the gateway peers.

The general concept in our system is that in order to provide this set of services, we have a number of discrete wireless sensor networks, each of which is connected to a certain set of control stations. These control stations act as Gateway peers in a peer-to-peer network and publish their own sensor network related data to the rest of the peers.

**IV. ARCHITECTURE OF THE PEERS - SERVICES PROVIDED**

The structure of the peers of our system follows a modular architecture of essentially two layers: the inner layer that is comprised of a minimum set of core functionalities and an outer layer, that hosts a variety of services. Figure 2 depicts the architecture at a high level.
The inner layer of the peers can be viewed as the kernel space of a microkernel architecture. Essentially it defines the domain model of the system and a minimum set of functionalities for controlling the entities. In analogy, the outer layer can be viewed as the user space of a microkernel architecture where services execute and communicate with each other via the inner layer.

A. The Inner Layer - Core Services

The inner layer of the peers maintains every information related to the structure of sensor networks, the specification of the devices, the results of the queries and every other information about data flow inside the network. It is the backbone of the peer as it serves a base for all the higher layer services to be deployed on. The entities that define the domain model of the system are categorized based on the services of the higher layers that use them as follows:

- **State** – entities that model information regarding the technical characteristics of the nano-peers, topological information of the physical network, configuration settings of the peers and operating parameters.
- **Geo** – entities that represent the coordinates and all the properties which assist in locating the nano-peers as well as points of interest connecting nano-peers and the physical world.
- **Queries** – entities that model information related to active and previously issued queries and the corresponding measurements received from the nano-peers.

Core services essentially offer a basis for using all the different kinds of information related to the system. As mentioned previously, it is important to be able to extend our system in simple ways, including tasks such as defining new sensors used by a new type of sensor nodes, or use thousands of nodes that create huge datasets in the system, and manage all this information in simple and efficient ways. First of all, we point out that in wireless sensor networks the originating location of data is crucial, and so there are a number of relevant entities organizing such information, based on a point-of-interest concept.

Regarding the extensibility of the system, a large degree of detail goes into defining the type of hardware used, the network organization and which node belongs to what network. A basic concept is “capability”, that is used to describe nodes’ abilities regarding their sensing and actuating capacities. Such capabilities are grouped in generic sets like “temperature”, “humidity”, etc. There is also the grouping in sensor networks and nodes.

There is also a set of managing entities (data managers) associated with the core services. These managers offer a set of functions in order to manage the core services entities and are implemented as singletons, based on an abstract manager entity, updating when needed the relevant entries in the system.

B. The outer layer

Outer layer services, as depicted in figure 2, are distinguished in 4 categories: a) peer-to-peer (p2p) services, b) WSN
platform services, c) persistence services, and d) web services and other interfaces. All the services offered by the system are implemented using Java.

Regarding p2p services, which are detailed in section V, our system uses JXTA to build an overlay network, upon which the unification of all sensor networks is based. In short, JXTA is a set of protocols and specifications for building systems that communicate in a p2p fashion. Even though JXTA originated from SUN, there are quite a few implementations, e.g., in Java (SE and ME), C, C++, C#, etc., that are also compatible with each other, thus being platform independent. Peers create an overlay network, through which each peer can communicate with everyone. Using JXTA peers exchange messages that describe their capabilities, query definitions and results from the sensor nodes, and so forth. The format of the advertisement messages exchanged between peers can be seen in figure 1. Regarding authentication, [19] describes the authentication scheme used in the WISEBED project (with Shibboleth playing a key role), which we intend to use also for our system.

Moreover, regarding the persistence services of the system, we chose to use Hibernate, which is an implementation of the Java Persistence API. Such an implementation, aims at offering an interface between an RDBMS and an application written in Java, in such a way that the developer is able to handle all data in the database as pure objects. Mapping between object-oriented features such as inheritance or polymorphism, and database tables and relations are handled by Hibernate. It automatically produces the relevant parts of the application code and acts as a middleware between the application and the database removing connectivity and updating complexity.

The WSN Platform services provided by the system essentially offer interfaces for using a variety of hardware and software technologies on the sensor network level, i.e., different types of sensor nodes and sensor boards. A description of the implemented drivers is provided in section VII.

Web services essentially define interfaces for users and applications to connect to our system and use the implemented set of services. A detailed discussion on a graphical user interface based on Google is provided in section VI.

V. Peer-to-peer Services

In this section we outline the peer-to-peer services provided by the system.

A. Buffering services

Wireless networks often face network disconnections, due to several reasons (poor quality links, environmental noise, etc.). The nodes of the overlay network handle these intermittent network disconnections using some kind of buffering service. This buffering scheme applies to all data, i.e., both data and control messages. Also, this buffering applies to data that needs to be forwarded to another node of the overlay network as well. For example, assume that a node of the network wishes to get some information from another node in the network that collects information from a wireless sensor network. At some point after they start communicating, the connection between these two nodes is temporarily disrupted. The second node will hold its information and wait until the network connection is restored, then it will start transmitting again.

B. Gateway for environmental monitoring & actuator control

Wireless sensor and actuator networks can be deployed for local monitoring and control of micromechanical devices. Scattered sensor devices have the capability to collect data. This high-level service provides access to such wireless networks and acts as a controlling center where data are routed to. The service is responsible for the gathering of all the readings
coming from the sensor networks and the forwarding of the queries from the data tier to the devices. In other words, they act as gateways between the wireless sensor networks and the fixed part of the global computer.

The queries on sensor networks may aggregate data over a group of sensors or a time window, contain conditions restricting the set of sensors from contributing data, correlate data from different sensors and trigger activation of an actuator. A user interacting with this service will typically issue a sequence of queries in order to obtain all the necessary information. Distributed query execution is optimized for both resource usage and reaction time.

On the hardware level, a typical service provider consists of one wireless device connected to a desktop PC or laptop along with a network connection to the global computer. The wireless device attached to the gateway is necessary to communicate with the wireless network. Alternatively, an embedded platform can be used. Data retrieved from the 40 wireless network are stored in a relational database fashion, with tables organized in three categories:

1) Device-related tables that are used to store information regarding the technical characteristics of the network nodes. We aim to support heterogeneous networks, i.e., networks that consist of different kinds of devices (e.g., TelosB, MicaZ motes etc.) which in turn may have different kinds of sensors and actuators attached to them (e.g., light, pressure, humidity, temperature etc.).

2) Query-related tables that keep track of active and historic queries. The service offers different type of queries that can be made by the user to the sensor and actuator network, and each query may refer to multiple devices and multiple sensors.

3) Sensor readings and actuator state tables that store the information received from the wireless network. Each record represents a reading coming from a specific device in the network concerning a single sensor or actuator.

C. SQL-like functionality

From a data storage point of view, one may think of a sensor network as a distributed database that collects physical measurements about the environment, indexes them, and then serves queries from users and other applications external to or from within the network. This service is responsible for the organization and storage of data after sensing actions, how program interfaces to the sensor database look like and how queries are processed and served in an efficient manner. The advantage of the database approach is that it provides a separation between the logical view (naming, access, operation) of the data held by the sensor network and the actual implementation of these operations on the physical network. Diverse sensor network users and applications can focus on the logical structure of the queries they intend to pose and are relatively isolated from the details of physical storage and data networking on the volatile physical infrastructure of the network.

The queries on sensor networks may aggregate data over a group of sensors or a time window, contain conditions restricting the set of sensors from contributing data, correlate data from different sensors, trigger data collection or signal processing on sensor nodes and spawn subqueries as necessary. A user interacting with a sensor database will typically issue a sequence of queries in order to obtain all the information he/she wants. Distributed query execution is optimized for both resource usage and reaction time. Initially, the service will support five basic SQL operators: count, min, max, sum, average. These basic functions will be extended to support more sophisticated data analysis.

D. Network statistics & management / control

Monitoring the condition of a wireless sensor network based on gross network metrics and also nano-peer local state is crucial for achieving good overall performance (i.e., at the application level) and also necessary to draw conclusions on the current state of the execution of a distributed algorithm. Also, we wish to allow the network administrator to adjust the operation of the network by modifying the parameters of the underlying protocols (e.g. cluster sizes, sleep schedules, encryption level etc.). The service collects information related to the operation of the nano-peers (e.g. information on the topology, available energy, neighborhood sizes, etc.), and allow the developer to program the network’s performance to match the application’s needs by modifying the operational parameters of the system. Such information can be used to further investigate the performance of the network and possibly draw conclusions on the current state of the execution of a distributed algorithm. The service allows to control the operation of the network by sending control messages to the wireless nodes. The control may be limited to the energy saving specification of the devices (e.g., the percentage of sleeping period) or extended enough to cover network operation parameters for the size of clusters, the rotation frequency of cluster heads, the time synchronization accuracy, etc. Depending on the level of details provided, the information can be used to execute an offline algorithm (e.g., resource management, scheduling, assignment) in order to fine-tune the performance of the network.

The collection of the status of the network is based on constructing global snapshots either periodically or on demand. Depending on the level of details provided, the information can be used to execute an offline algorithm (e.g. resource management, scheduling, assignment) in order to fine-tune the performance of the network. The control may be limited to the energy saving specification of the devices (e.g. the percentage of sleeping period) or extended enough to cover network operation parameters for the size of clusters, the rotation frequency of cluster heads, the time synchronization accuracy, etc. Related functionalities: Support management, Mobility prediction & control, Energy management/topology control, Time synchronization, Clustering/grouping, Data propagation & query dissemination
E. Virtual sensor networks

This service offers the ability to manage multiple wireless sensor networks, each with a different control center, under a common installation. This is done by introducing the notion of a virtual sensor network that hides the actual network topology and allows the user to control the motes as if they were deployed under a single, unified, sensor network. This abstraction significantly reduces the overhead of administering multiple networks. Furthermore, the idea of a unified, virtual sensor network allows the integration of totally heterogeneous sensor networks, i.e., not only regarding different kind of sensors attached to the motes of the network, but also different kind of CPU architectures that communicate under different RF and optical devices.

For each sensor network, a unique ID is given to its respective control center. This sensor network ID helps to distinguish one mote from another, when they have the same mote ID but belong to different sensor networks, thus making it possible to manage different networks in a unified way.

VI. END-USER INTERFACE WITH THE SYSTEM

In this section we briefly describe the end-user interface and the visualization of results of our system through the Google Earth application. Google Earth is a freeware application that provides 3D representations of geographic as well as other kinds of information, through a representation of Earth. It is available for several operating systems, while also a browser plugin has been made available, essentially allowing its integration into popular browsers like Firefox. Apart from the advanced 3D visualization capabilities offered by this application, it uses a simple and elegant model for representing the information used by the application, allowing for programmers to utilize it in order to extend other software systems and offer a new user interface through Google Earth, with all its impressive features. This is the precise idea behind the interfacing of our system through Google Earth, i.e., to create a user interface based on the visualization capabilities offered by this application.

In short, all information that is displayed through Google Earth is expressed in KML (Keyhole Markup Language). KML is an XML dialect for representing geographic information in 3D virtual environments such as Google Earth, or it can be used in other applications (such as Google Maps). It follows a tag-based structure with nested features and attributes, based on a strict model. Through KML we can express geographic information, 3D components (e.g., buildings), time-evolving information, etc. We use KML to represent entities such as:

- The nodes of the wireless sensor networks that participate in the peer-to-peer network.
- The buildings that these nodes are located in.
- Data representing the measurements from the system’s sensor nodes.
- Representation of the communication links between sensor nodes.

We can organize all these different sources of information in order to present a unified interface to the end-user. The general idea is that through such an interface users can query the system for the whole or part of the data produced by the peers through Google Earth. As mentioned before, the sensor networks and their nodes that participate in the system are linked to specific geographic information, that either describes the overall area that they are currently situated in or their precise location (longitude, latitude, altitude). Such information can be combined with 3D representations of the responding buildings in order to visualize the sensor networks in a 3D space. Building representations are provided by the users participating in the peer-to-peer network, i.e., they are not automatically generated by our software, using the suitable tool (i.e., Google Sketchup).

This specific feature, i.e., the display of sensor networks combined with 3D building models, sets it apart with respect to the user interface, since most similar environments use 2D representations. It offers the advantage of the simultaneous display of multiple sensor networks, that is especially useful in the case where we have multiple sensor nodes or networks spanning across a single building. An example of such an installation can be seen in Figure 7, where the different sensor networks deployed in the Research Academic Computer Technology Institute building and spanning across 4 different levels, are depicted. Such representation are also potentially useful in the case of structural health monitoring applications, where 3D visualization can give an instant overall view of the conditions inside the building.

Regarding the actual implementation of the user interface (which is still under development), in short, we currently use a number of Java Servlets that are executed inside an application server, that can be located in the user’s computer, or it can be a special peer in the system acting as a producer of such representations. These servlets are using the services described in the previous sections of this paper, in order to have access to the features of the system and the data coming from the sensor networks, and combined with a number of pre-made 3D models, they produce output in KML form that can be accessed by Google Earth through the use of a network link. Such links point to specific URLs, that represent the connection with the application server.

Regarding the structure of the information displayed to the end-user, we make the distinction between two kinds of information:

- the 3D information displayed in the main application window,
- the information regarding the features of the system that are displayed in a tree-wise form in the application’s sidebar.

Main application window: in this part of the screen information that includes the sensor nodes, the respective measurements and the buildings where these nodes are located, is displayed in 3D form. Sensor nodes are displayed in placemark form: a specific point in the virtual space is assigned to each sensor belonging to a sensor node of the system (according to its actual location) along with an icon representing the...
sensor (e.g., for a temperature sensor a thermometer icon is used). Users can select with the cursor the placemark for each such sensor, and as a result a new window with information regarding this sensor and sensor node will prompt up in the screen, e.g., the timestamp and value of the last sensor reading, network statistics, etc. Moreover, the latest values of the measurements provided by the sensors of the system are displayed in a 3D bar form, that is located in the sensors’ coordinates and is colored according to predefined values (e.g., a red bar means temperature over a certain threshold). Also, the routing tree setup inside the sensor network is depicted a series of arrows, according to the actual links between sensor nodes.

Application sidebar: the sidebar displays information as a drop-down list, where the upper layer is the discrete sensor networks, the middle layer is comprised by the sensor nodes and the bottom layer includes the discrete sensors belonging to the system. The user can use the sidebar in order to move from one sensor network participating in the system to the other, i.e., the main window view moves from one geographic area to the other.

Specific examples of the functionality provided by the user interface can be seen in Figure 7.

VII. WSN PLATFORM SERVICES

In this section, we give a brief overview of the provided WSN platform services. Essentially, we will describe the concepts used in interfacing a software system running on a sensor network level (i.e., on the nano-peers) with the rest of the system.

First of all, we point out that generally the design of the respective services follows the Observer and Controller software design patterns. This basically means that a set of data producers and consumers are used to exchange data between the different levels of the system, and some controlling authorities are mediating in order to synchronize these entities. A Controller/Observer entity is used as a basis for these controlling authorities, e.g., in the case of reading messages produced by the sensor network nodes and handed over to the system by the network gateways a ReadingController is used. The general concept used is the following:

- Entities that produce Readings (see below) hand over the new Reading objects to the ReadingController.
- Entities that consume Readings (i.e., the part of the system that logs readings to a system database) observe the ReadingController and get notified (call-back) when a new reading arrives from any sensor network or by other system peers.

The specifics of the interconnection with the sensor gateways and the translation of incoming and outgoing messages are handled by a set of Drivers, that implement the relevant functionality, depending on the software system the sensor nodes use. We currently use three such systems:

- Octopus [20]: Octopus is a TinyOS 2-based software allowing for a number of basic operations on the sensor nodes, including querying sensors, setting sleep/activity periods, etc., using a simple GUI. We use a modified version of this system, which includes some extensions like support for additional types of sensor nodes, including MicaZ, TelosB and a number of sensor boards, e.g., MTS310CB, MDA100. The latest version also supports the definition of user alerts when certain events occur inside the sensor network, such as when temperature reaches a certain threshold, etc.
- Dissense: This system is used by the university of Rome to monitor their Their Sky-based (TelosB) testbed site. It is developed using TinyOS 2 (like Octopus) and focuses on the least energy used possible (see section VIII-B2 for more details).
- XMesh/XServe [21]: XMesh is a firmware provided by Crossbow (part of the Moteworks software suite), enabling the polling of sensors and the formation of a multi-hop network. The Moteview GUI is used to administer these nodes, while the XServe software running on the gateway stations allows for streaming XML communication from and to the sensor nodes.

We also use a basic Java component for the Sun SPOT-based parts of the testbed (see section VIII) polling the embedded temperature and light sensors every five minutes and sending the reading back to a base station. In the current incarnation of the system, there are 3 relevant Drivers implemented:

- TinyOSDriver: this driver essentially provides an interface to a Serial Forwarder entity, that mediates between TinyOS-based sensor networks and the rest of the world. For each message type used inside the sensor network, a certain handler has to be defined which is also registered as a message listener.
- SpotDriver: this driver provides interconnection with a Sun SPOT gateway station. Sun SPOTs use a JavaME-compliant virtual machine, so the interconnection with the rest of the system is straightforward.
- XServeDriver: this driver connects to a gateway station running XServe. XServe provides XML streams for receiving readings and tasking the networks, and the driver uses this feature to provide interconnection with the rest of the system.

VIII. DESCRIPTION OF THE INSTALLED TESTBED

In this section we describe an experimental testbed that was operational for a time frame of several weeks in October and November 2008. Overall, 5 different research teams took part in the testbed, from:

- the University of Patras, Greece.
- the Research Academic Computer Technology Institute, Greece.
- the University of Ioannina, Greece.
- the SAPIENZA University of Rome (UDRLS), Italy.
- the University of Geneva, Switzerland.

The purpose of these sensor networks was to monitor a number of conditions inside the buildings that housed these
specific networks. In the majority of cases we were interested in monitoring temperature, light and relative humidity inside these areas. Also we were interested in confirming in reality that our system possesses basic features described above, such as the extendability, the potential of creating large sensor networks and the ease of use of the system overall. We were able to monitor in great detail the conditions inside these buildings, in order to use them in the future for realizing an energy-saving building policy. As stated in the introduction, one of the major goals of WebDust is to support heterogeneity in both hardware and software. As a result, we used a quite diverse set of sensor nodes, gateway stations, and software running on a sensor node level. We provide a detailed description of our hardware and software setup.

A. Hardware overview

Regarding the hardware used in the testbed, a total number of 56 nodes was used in our testbed, with 4 different types of sensor nodes (mica2, micaZ, TelosB, SUN Spot). TelosB and SUN Spot nodes have integrated temperature, light, humidity and temperature, light, acceleration sensors respectively. We had to use separate sensor boards for mica2 and micaZ nodes. In particular, we used the MTS310CB, MTS400CB, MTS420CC and MDA100CB sensor boards provided by Crossbow. A more detailed description of each component can be seen in table I.

Overall, the testbed comprised of 9 discrete sensor networks:

1) At CTI’s data center room, comprised of 4 TelosB nodes, measuring temperature, light and relative humidity.
2) Floor 0 at the RACTI building, comprised of 10 TelosB nodes (temperature, light and relative humidity).
3) Floor 3 at CTI’s building, comprised of 6 SUN Spot nodes, measuring temperature and light.
4) Floor 4 at CTI’s building, comprised of 6 mica2 nodes, measuring temperature, light, relative humidity, barometric pressure, magnetic fields.
5) At the graduate students’ offices of the Dpt. of Comp. Eng. and Informatics, University of Patras, we used 5 mica2 nodes measuring temperature and light.
6) At the main building of the Dpt. of Comp. Eng. and Informatics, University of Patras, we used 4 mica2 nodes measuring temperature and light.
7) At the building of the Computer Science Department at the University of Ioannina, we used 6 mica2 nodes measuring temperature, light and relative humidity.
8) At the University of Rome we used 11 TelosB nodes, measuring temperature, light, and relative humidity.
9) At the University of Geneva we used 4 MicaZ nodes, measuring temperature, light, magnetic fields.

Regarding the gateways used for each sensor network, we used a number of Crossbow Stargate Netbridges and MIB600 network programming boards, for the mica2, micaZ and TelosB sensor nodes, together with a number of Alix gateways for the SUN Spot nodes. Alix stations use an AMD Geode CPU at 500 MHz, have 256MB of RAM and 4GB of Compact flash as main storage, 2 USB ports and WiFi as an option. Their size is relatively small (Mini-ITX) allowing for us to embed them easily in building offices and combined with their low consumption (4W), Alix is an ideal platform to use as a sensor network gateway running untethered year-round.

B. Testbed architecture overview and software used

Regarding the software used in our experimental testbed, we used WebDust as the core of the system. As mentioned previously, WebDust is independent of the software running on the sensor network level, as long as a suitable software “driver” is used to communicate with the gateway stations of the responding sensor networks.

Each of these individual campus networks (Patras, Ioannina, Rome, Geneva) is represented in the overall system by a software peer running WebDust, that can send and receive readings from and to the rest of the peers. We will now refer to some details about the deployment of sensor nodes in the RACTI building in Patras and in the University of Rome.

1) CTI Deployment: Essentially, the deployment follows the layout of the respective buildings. E.g., in RACTI every level is divided into two or three sectors, that are somewhat isolated with regard to wireless communication between them, due to thick cement walls and metallic doors. For this reason, there has to be a sensor gateway in each sector of the building for interfacing with the rest of the testbed. Moreover, we chose to use wall-mount plug adapters to power all sensor nodes in the RACTI building and not use rechargeable batteries, due to practical difficulties we faced during the operation of the testbed (mainly changing batteries and problems with the nodes’ availability). As mentioned previously, the sensor networks located inside the RACTI building span across 4 different floors, and cover a significant percentage of the total building area.

On a sensor network level, for the networks that are comprised of mica2, micaZ and TelosB nodes, we use mainly TinyOS as a base for our system. The nodes are programmed with a modified version of the Octopus environment (see section VII). A small number of nodes is programmed with the XMesh firmware provided by Crossbow (also section VII), mainly due to the lack of proper support in TinyOS 2 of the Crossbow MTS400 series of sensor boards. We “linked” the Octopus and Xserve gateways to our WebDust peers by writing a simple driver, that basically interprets messages coming both ways.

In a similar manner, we had to interconnect the sensor networks comprised by SUN Spot nodes to our system. These devices use a specialized Java virtual machine called SquawkVM, that is fully J2ME-capable and is also used as a kind of operating system. As a result, all relevant code is written in Java, making it easier to interconnect the gateway with WebDust.

Regarding the software used in the gateways, we use the Xubuntu Linux distribution in Alix gateways, mainly due to the fact that it provides a lightweight complete environment for development and use. A special Debian distribution (provided
by Crossbow, running on a USB flash disk) is used in Stargate NetBridge gateways. This distribution contains the Moteworks middleware that is used together with the XMesh sensor nodes’ firmware.

2) UDRLS Deployment: During the renovation of the buildings of the Department of Computer and Systems Sciences at SAPIENZA, roman remains dating between the 2nd and the 4th centuries A.D. were found and activities have started for the protection and musealization of the site. In particular, in cooperation with the Soprintendenza Archeologica di Roma, we have started an activity based on a Wireless Sensor Network for monitoring the environmental parameters of the archaeological site: we measure temperature, humidity and light which are connected with the growth and development of fungus and mildew [22] and may damage painted surfaces [23]. The considered site has only modest and small frescoes and it is entirely made of stones, travertine and bricks, which are not harmed by fungus and mildew, but in any case the aesthetic consequences might be unpleasant. Furthermore archaeologists were interested in a testbed to verify the effectiveness of WSNs in monitoring archaeological sites. WSNs are particularly well suited for this purpose, they are wireless ad-hoc networks made of tiny nodes and thus neither wires nor bulky instrumentations have to be deployed over the ancient artifacts.

Most of the research on sensor networks is focused on the design of very large and dense networks often subject to "strict" latency constraints, but many practical applications, such as the ones presented in this paper, consider relatively small networks. In particular our testbed is made of 12 battery-powered TelosB nodes running TinyOS and equipped with temperature, humidity and light sensors: half of them are deployed in the about 16 square meters of the archaeological site, while the others form a multihop network to connect the site with the gateway located in a room distant about 40 meters. Archaeologists were not interested in real-time observations, a sample per hour for each physical quantity reported every 12 hours can be sufficient, rather they were interested in maximizing the duration of the monitoring activities (i.e. network life time).

These requirements motivate the design of a very simple, but practical energy efficient protocol, capable to guarantee a network life time of approximately 3 years. Nodes alternate periods of activity, during which there is a significant energy consumption, and inactive periods during which energy consumption is minimized. During the inactive periods, nodes power off their radio controller but still sample the environment at regular intervals specified by the sampling rate; sampling is an order of magnitude less energy consuming than all activities involving the radio. During the active periods, all nodes must turn on the radio and a 6-phase protocol is then executed:

- **phase 0.** A guard period (which takes into account node’s clock drifts) is used to ensure that all nodes are awake at the beginning of the next phase;
- **phase 1.** A routing tree is generated. In our case we use the CTP protocol\(^1\), but any routing algorithm can be adopted;
- **phase 2.** An alive message is sent to the sink from all the nodes;
- **phase 3.** The sink replies with a Data Request Message. This message is also used to re-synchronize the clocks and to communicate the next scheduled active interval;
- **phase 4.** All nodes send the sampled data through the routing tree;
- **phase 5.** All nodes start a new inactive period and schedule their next active period.

The gateway is a server running XubunTOS Linux distribution with the sink connected to an USB port. A Java software has been developed allowing bidirectional communication between the gateway and the sink: it stores data readings on a database and allows us to update the sampling rate and the scheduling intervals. The collected data are available via a web interface. It is possible to specify a time interval and observe the corresponding data displayed on a chart. Furthermore, it is also possible to obtain statistics on the data (e.g., average

\(^1\)http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIB600</td>
<td>Crossbow sensor node programming board, with Ethernet connectivity</td>
<td>2</td>
</tr>
<tr>
<td>Stargate NetBridge</td>
<td>Intel IXP420 Xscale CPU at 266MHz, 32MB RAM, 2GB USB Flash Disk</td>
<td>2</td>
</tr>
<tr>
<td>Alix</td>
<td>AMD Geode CPU at 500MHz, 256MB RAM, 4GB Compact Flash Disk</td>
<td>2</td>
</tr>
<tr>
<td>MTS310CB</td>
<td>Sensor board with temperature and light sensors, microphone, accelerometer,</td>
<td>8</td>
</tr>
<tr>
<td>MTS400CC</td>
<td>Sensor board with temperature, light, humidity, barometric sensor and</td>
<td>8</td>
</tr>
<tr>
<td>SUN Spot</td>
<td>32-bit CPU, 166 MHz, 512KB RAM, 2.4GHz IEEE 802.15.4 RF Chip</td>
<td>6</td>
</tr>
<tr>
<td>TelosB</td>
<td>16-bit CPU, 8 MHz, 10KB RAM, 2.4GHz IEEE 802.15.4 RF Chip</td>
<td>25</td>
</tr>
<tr>
<td>micaZ</td>
<td>8-bit CPU, 8 MHz, 3KB RAM, 2.4GHz IEEE 802.15.4 RF Chip</td>
<td>4</td>
</tr>
<tr>
<td>mica2</td>
<td>8-bit CPU, 8 MHz, 4KB RAM, 900MHz RF Chip</td>
<td>21</td>
</tr>
<tr>
<td>Sensor Nodes</td>
<td>32-bit CPU, 166 MHz, 512KB RAM, 2.4GHz IEEE 802.15.4 RF Chip</td>
<td>25</td>
</tr>
<tr>
<td>MTS420CC</td>
<td>Same as MTS400CC with additional GPS</td>
<td>1</td>
</tr>
<tr>
<td>Proteotyping area board with temperature and light sensors</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Sensor Boards</td>
<td>MTS400CB</td>
<td>Sensor board with temperature and light sensors, microphone, accelerometer, magnetometer</td>
</tr>
<tr>
<td>Gateways</td>
<td>MDA100CB</td>
<td>Prototyping area board with temperature and light sensors</td>
</tr>
</tbody>
</table>

TABLE I

LIST OF THE HARDWARE USED IN THE TESTBED
temperature) and data can be filtered out by node id and data type.

The WSN is fully integrated with WebDust by redirecting the information flow to a WebDust peer. A 3D model developed with Google Sketchup is already available enabling WebDust to display the site’s data on Google Earth. Recently we have started a new activity to control the site’s ventilators by the WSN itself.

IX. CONCLUSIONS AND FUTURE WORK

In this work we have presented the architecture and aspects of the implementation of a system interconnecting multiple lesser-scale wireless sensor networks into a virtual large-scale network, based on a peer-to-peer overlay. This system can also act as a part of a grand-scale overlay network, that appears as a “global computer”, offering a wide range of resources and services to the users. We presented the use of our system to unify 5 discrete sensor networks amounting to a total of 56 heterogeneous sensor nodes, using a set of different software subsystems on the sensor network level. Future work on this system includes extensions to the user interface, integration with actuator devices, as well as with other software environments specifically for sensor networks for providing additional features to end-users.

ACKNOWLEDGMENTS

This work has been partially supported by the IST Programme of the European Union under contract number IST-2005-15964 (AEOLUS) and by the ICT Programme of the European Union under contract number ICT-2008-224460 (WISEBED).

REFERENCES


Fig. 4. A TelosB node plugged to a wall-mount adapter

Fig. 5. A SUN Spot node next to an Alix gateway

Fig. 6. A screenshot showing routing tree representation in Google Earth (UDRLS testbed)

Fig. 7. A screenshot of the Google Earth part of WebDust’s interface (CTI Patras Testbed)
TinyAID: Automated Instrumentation and Evaluation Support for TinyOS

Christoph Weyer, Christian Renner, and Volker Turau
Hamburg University of Technology
Institute of Telematics
Schwarzenbergstraße 95
D-21073 Hamburg, Germany
{c.weyer,christian.renner,turau}@tu-harburg.de

Hannes Frey
University of Paderborn
Computer Networks Group
Pohlweg 47-49
D-33098 Paderborn, Germany
hannes.frey@uni-paderborn.de

Abstract—TinyAID is a tool that supports automated instrumentation and evaluation of TinyOS-based distributed applications. Two types of instrumentations are provided: logging of call chains and message flows within the network. TinyAID assists the debugging process by post evaluation of the logged data. A main benefit is the visualization component for representing traces in their spatial and temporal order.

The instrumentation and evaluation concepts are evaluated in two case studies: the SelfWISE framework and a selection of routing algorithms. Due to the automated process of TinyAID the evaluation could be performed without a deeper knowledge of the implementations under test. In the first case TinyAID revealed a weakness in the TOSSIM random number generator. The second case demonstrates the power of TinyAID to visualize the quality of protocols in a unified manner, without any manual changes to the specific source code.

I. INTRODUCTION

Sensor networks consist of small, micro controller driven wireless network nodes with additional sensing capabilities. Once deployed in a certain environment, such nodes are supposed to set up a wireless network in an ad-hoc fashion and to run unattendedly for a long period of time. Individual sensor nodes collect data about a certain physical phenomenon. The gathered data can, by means of the wireless network, be easily moved towards one or more data collection nodes.

Sensor networks are a promising approach. In the future they may support measurements in application domains that will be far beyond what can be measured today. The general idea is appealing and attracted many researchers. A lot of research has been conducted on protocol design for various problems, such as data communication, topology control, time synchronization, data collection, tracking, or activity scheduling, just to mention a few. Once a new protocol has been developed and some positive properties have been investigated, theoretically or by means of simulation, producing an executable for real sensor networks still requires a lot of additional effort.

We see that there is a high demand for supplemental development support. One important goal is to support protocol debugging, i.e., a common mechanism to test correctness of and to identify errors in existing protocol implementations. Another important goal is to devise a protocol evaluation system. Such a system would provide a common base to compare different protocol implementations with each other. This may, e.g., be used to show the benefits and drawbacks of a new implementation as compared to existing protocols. It would also allow for investigating the improvements of a new version over an existing one.

In this work we consider the problem of protocol debugging and protocol evaluation. While at first glance those two problem areas appear to have nothing specific in common, we argue in Sect. II that automated instrumentation and automated evaluation support is a striking methodology to address both problems. The term automated instrumentation describes the process of automatically adding instrumentation code to an existing implementation, so that the extended code produces output usable for further analysis. The most important kind of output in this context is logged data, i.e., producing data sets that systematically encode the protocol behavior and protocol states in a chronological order. These data sets can further be investigated offline. This process is called the post evaluation pass. Investigation of the data sets may be performed manually or in an automated way.

From our perspective performing instrumentation automatically requires a framework and coding conventions for the development of the protocols under investigation. Only if the code complies with this, an automatism can find the right hooks to include the appropriate instrumentation code. We believe that TinyOS – having a large community and already consisting of a multitude of relevant protocol implementations – will benefit from automated instrumentation support. In Sect. III we suggest a simple yet effective way for automatic instrumentation of TinyOS-based protocol implementations.

The most relevant elements of evaluation support we see here are tracing and statistical analysis. Tracing refers to a representation of the behavior of the system under test with respect to temporal or spatial properties. With statistical analysis we refer to all mechanisms that analyze aggregations of logged data. Processing and evaluating large amounts of data is achieved by information visualization that leverages the human visual capabilities, thus allowing to detect complex or even unexpected interrelations. In Sect. IV automatic evaluation examples from both classes are presented.
This work is not considering a specific protocol, but introduces a general concept for protocol instrumentation and evaluation. Assessing such a concept is a less common case and some characteristics are hard to quantify. However, we provide partial answers on this difficult question in Sect. V. To test and illustrate usability of the suggested concept, it is applied to existing applications and routing protocols. We also investigate the ease of use of automated instrumentation, the obtainable results from automated evaluation, and the overhead incurred when applying these concepts. In Sect. VI an overview of other existing tools is presented. Finally, Sect. VII will provide a preview on the necessary steps ahead to finally implement our envisioned tool for real deployments.

II. PROBLEM STATEMENT

A. The Need for Debugging Support

Programming usually requires gradually improving a first version towards a final release that offers reliable correctness to a considerable extent. For single-threaded programming the use of breakpoints is the primary choice. However, this becomes ineffective, if applied to multi-threaded or even distributed programming.

In principle it is possible to extend the code for a sensor node with breakpoints, for instance by using the JTAG port of the micro controller. In a deployed system, however, this approach is like removing that node from the sensor network. Reaching a breakpoint disables all interrupts. Hence, the node will not react on any incoming messages or on timeouts. Thus, the inspected system may behave highly different from an un inspected one. Moreover, due to the non-deterministic behavior of system parameters, such as clock drift, message delivery success, or message transmission times, a system may behave differently in each evaluation pass.

We believe that logging support is the most appropriate solution here. It is capable of providing a fine-grained chronological list of state information of every node. Once established, the log can be used in a post evaluation pass for further analysis.

However, adding instrumentation code to existing software has a few drawbacks. The additional code changes the runtime behavior. Especially in critical parts of the source code, e.g., interrupt handlers, the insertion of code can lead to a change in the system behavior. Therefore, the places and the quantity of the instrumentation code must be carefully chosen to keep the influence on runtime behavior as small as possible.

B. The Need for Generic Evaluation Support

When comparing existing protocols with a new protocol implementation, common metrics are needed. Using manual instrumentation for obtaining measurements and computing these metrics causes problems. While adding code lines for logging to an own implementation may be easy, supplementing foreign code with them requires some effort: that implementation has to be studied and understood in the first place. However, necessary places for adding logging code might be overlooked both in the own and particularly in the code of others. As a result, a share of message transmissions will not be available in the message logs, most likely leading to misinterpretation of the protocol characteristics. Having an automated way of instrumenting code will solve this problem. Such a mechanism would add logging code, e.g., at places where messages are transmitted.

C. Limitations

While we propagate automated instrumentation as the primary choice for protocol debugging and comparison, we also see limitations on what can be achieved with that concept. Consider for example an automated message logging feature producing a file containing information about time and visited nodes of certain message instances. Say we want to investigate the success rate of a given single path routing protocol. We can investigate the message type that is used to transmit data with that protocol. Every message instance of this type appears on a certain node, visits a sequence of nodes along the routing path, and finally disappears. However, using such a log file, how can we know that the last visited node was the message destination and that the message was not just dropped due to a routing failure?

We can’t. One has to introduce additional coding conventions, e.g., the use of a certain routing framework with generic routing functions for sending a message to the next hop, dropping a message on failure, and consuming the message on success. In this case information about those function calls could be exploited to infer delivery failure and success automatically. However, if we want to deal with legacy code, inventing such additional coding conventions is not an option.

Another important example is a state machine implemented by a large switch statement within the code block of a single module. Depending on the state, a certain part in the switch statement is executed. Since the effects of two states may be exactly the same, despite a local variable encoding the state, it appears to be impossible to infer information of state changes without having some additional knowledge about the code semantics.

It follows that while we can find many application cases where automated instrumentation is a productive tool, there are those cases where only partial information can be produced automatically. Not to exclude those cases, in which additional means for manually adding application context information into the instrumentation and evaluation process are required. For instance, in the routing example discussed above, application context could determine the transport layer end points which were served by the investigated routing protocol.

III. TINYAID INSTRUMENTATION

TinyAID currently supports two kinds of automated code instrumentation: call-chain logging and message logging. Figure 1 depicts the tool chain of automated code instrumentation. Given any nesC source code, the TinyOS tool chain first creates a single, plain C file by combining this code with the TinyOS components used. The automated code instrumentation intercepts the TinyOS tool chain after this point, adding
an additional preprocessing step, called instrumentation pass. Given a certain configuration file, “config.cfg” in this example, the instrumenter inserts additional instrumentation code, provided by a code template, into the plain C file. The code template reflects the way how the log information is dumped on a given target platform. This step results in an instrumented C file that will then be handed back to the remaining TinyOS tool chain. Depending on the target platform, an instrumented program image or a TOSSIM library, in case of simulations, is finally created.

Fig. 1. Automated instrumentation by intercepting the TinyOS tool chain.

A. Call-Chain Logging

In call-chain logging the enter and exit times of certain event handlers and functions of the nodes are logged. This is achieved via additional code that is added to these handlers and functions during the instrumentation pass. For every handler and function, logging code is added immediately after the function entry point, at the end of the function, and immediately before each return. The wiring of TinyOS components is converted into C functions containing only a return statement with the next function call as parameter. These referring functions are skipped by compiler optimizations. Therefore, no instrumentation code is inserted in those functions. Every logged enter and exit event constitutes a single line. Data written in a line is the node ID, a time stamp, either > for enter or < for exit, and an identifier for the called event handler or function.

Since call-chain logging may result in very large data sets, the logged data only consists of unique integers. During the instrumentation pass, a separate file is created, which maps every unique event handler to a unique integer value and vice versa.

An extract of an example call-chain log file is depicted in Fig. 2. In this example, event handler 42 of node 5 is entered at time 1320 ms. Within this, a nested call to functions 36 and 12 is performed. Later at time 1684, event handler 20 of node 3 is called, returning immediately without any other nested calls. Finally, at time 1930, event handler 42 is called again, however, now on node 7. This time the handler is passed without any other nested calls.

Fig. 2. An extract of an example call-chain log file.

The event or function to be logged is defined by the configuration file used during the instrumentation pass. This information is defined by using regular expressions, which are matched against the unique C preprocessor event handler and function identifications in the plain C file created by the nesC compiler.

Every line in the call-chain configuration file starts with either ‘+’ or ‘-’ to include or respectively exclude event handlers or functions that match the following expression. The inclusion or exclusion symbol is followed by d, f, or h to decide whether the following regular expression is applied on directory names, file names, or handler and function names, respectively. This is followed by the regular expression.

The instrumenter steps through the plain C file and checks for every encountered function or event handler entry point, if they match any of the expressions of the configuration file. For this, the list of expressions is scanned from top to bottom, until the first match is found. Depending on the inclusion and exclusion flag, this line decides, if code instrumentation is applied or not. If no entry is found, the code instrumentation is not applied.

Refer to Fig. 3 for an example. The first line excludes any code residing inside the directory /opt/tinyos-2.x from being instrumented for logging. The next line demands that all event handlers and functions implemented in file Test.nc are instrumented. The remaining two lines include the event handlers fired and booted for instrumentation.

B. Message Logging

TinyAID also supports logging of information about messages that have been created, sent, or received by the Active Messaging framework [1]. This is obtained by the automatic instrumentation of the Active Messaging functions. Basically, additional logging code is added immediately after the entry points of the Active Messaging functions AMSGend.send, Receive.receive, and Packet.clear. For each supported platform, the configuration file has to provide the configuration file has to provide the
names of the modules implementing these functions. Thus, the instrumenter can automatically insert the code templates for message logging. It also extends the message header by a unique message ID. The latter consists of the address of the node having created the message (the message’s origin) and a sequence number. Each message is tagged with this information at creation time. Here we utilize the Active Messaging coding convention that for any newly created message the Packet.clear function has to be called. Hence, message tagging is completely transparent to the user.

The actual code for creating the log files has to be provided by the code template file. This file has to contain code snippets which are invoked on message creation, transmission, and reception. In our current example template file, the code snippet logs the following information: address of the message creating node, creation time, character c for encoding the message creation event, and message ID (origin plus seqno). For an example, refer to the first data entry in the example log file in Fig. 4.

<table>
<thead>
<tr>
<th>node</th>
<th>time [ms]</th>
<th>action</th>
<th>type</th>
<th>src</th>
<th>dest</th>
<th>origin</th>
<th>seqno</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3520</td>
<td>c</td>
<td>17</td>
<td>3</td>
<td>12</td>
<td>3</td>
<td>42</td>
</tr>
<tr>
<td>3</td>
<td>3521</td>
<td>s</td>
<td>34</td>
<td>5</td>
<td>65535</td>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>3535</td>
<td>r</td>
<td>34</td>
<td>5</td>
<td>65535</td>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>3520</td>
<td>c</td>
<td>17</td>
<td>3</td>
<td>12</td>
<td>3</td>
<td>42</td>
</tr>
</tbody>
</table>

Fig. 4. An extract of an example message log file

For message transmission and reception the corresponding code snippets provided in the code template are also passed. In our current example template file, both create a data entry with the same information as provided upon message creation. In contrast, however, send and receive events are distinguished by the characters s and r in the action column. In addition, the message type, the sender (src) and destination (dest) are also logged. Note that the message ID, consisting of origin and sequence number, can be used to relate packet creation and the origin’s final packet transmission, which could be far apart in the log file due to, e.g., message buffering in the routing layer. Furthermore, unicast and broadcast can easily be distinguished, because the latter uses a destination address of 65535. If node and destination address are not equal (and the latter is not the broadcast address), message overhearing can also be tracked.

Refer to Fig. 4 for an example. A message with ID 3-42 is created at time 3520 on node 3. At time 3521 it is then sent to node 12. This node is receiving the message at time 3535. In the mean time another message with ID 5-14 is created at node 5 and broadcasted. Finally, at time 3520, node 3 creates another message 3-43. From the sequence numbering it is clear that this is the next message after the above considered message sent from 3 to 12.

C. Manual Instrumentation

There are two main situations, in which manual code instrumentation may become unavoidable. These include identifying the visited states of certain state machine implementations, and secondly identifying the end points of communication protocols.

For the first aspect, a function state(name) is introduced. It can be added manually at any code line. The instrumenter will create a mapping from state names to automatically generated state IDs. Again, as with call-chain logging, the additional mapping is used to keep the logged data compact. Code execution passing such function will produce an entry in the call-chain log file. The direction will be denoted as !, and the column handler ID will be used to store the state ID.

For identifying correct delivery of message communication the function consume(msg) is introduced, which has to be added at those code places where semantically the message successfully reaches its destination. Whenever code execution passes that function, the message ID and its type are obtained from the given message. In the log data, message consumption is denoted by an x in the action column.

IV. TinyAID Evaluation Examples

The automated evaluation process is implemented by the TinyAID evaluator. As depicted in Fig. 5, it requires a set of call chain log and message log files in order to produce the appropriate evaluation files. An additional input file context.dat provides information about how the files are to be combined, which data has to be extracted, and which additional application context is required to evaluate the data. The output produced can be classified in statistics and visualization data. Statistics data are either data sets or PDF files. Visualization data are either PDF files or videos representing behavior over time.

A. Event Tracing

This concept is a pictorial representation at which time which node was entering a certain event handler. In order to produce such representation, the evaluator requires a single call-chain log file, the information about which event should be inspected for which nodes during which time period, and which resolution should be used for x- and y-axis. The result will be a PDF file with one or several lines of time line

Fig. 5. Automated evaluation based on call chain and message log files.
representations. The time is depicted on the $x$-axis and nodes are depicted on the $y$-axis. Every time a node enters the inspected event handler, a marker is placed on the time line of that node. Refer to Fig. 6 for an example of a single line event tracing result. Which information can be inferred from this figure?

![Event Tracing Example](image)

Fig. 6. An event tracing example.

We see several visual indicators which can be useful for protocol debugging. The first one is detection of event starvation. This term refers to an event that is expected to occur regularly – e.g., a regular beacon –, but suddenly happens to be missing. Such error might occur in a complex protocol where the scheduled timeouts might be deactivated in certain critical protocol states. If timeout activation on leaving of this critical state was forgotten in the implementation, the timeout may never occur again. For instance, in Fig. 6, starting at about time 1100, node 4 shows an indication of event starvation.

Another indicator that can visually be detected is that of event explosion. This term refers to an event that – compared to the event history – starts to occur with an abnormal high frequency at a certain time. Figure 6 depicts an event explosion example. Compared to its event history, at about time 1200, the observed event happens to occur unusually often on node 3. Such an error might occur, when a repeating timeout event handler is supposed to schedule the next timeout whenever it is called. The first time, however, this event has to be scheduled from somewhere outside the event handler code. If by implementation failure, code for scheduling the timeout event from outside the event is accidently executed a second time, two parallel lines of this repeating timeout event will be running on the node. Things are getting worse, if this failure happens to occur from time to time. Eventually the node will exclusively be busy with handling all of these timeout events.

Two other types of visual error indicators refer to the timing of events. One indicator which we call event jittering shows a burst behavior of event occurrences. An example can be found for node 2 in Fig. 6. At about time 1000, time 1100, and immediately before time 1200, an event accumulation can be observed while barely any other event occurs in between. This can be an indication of a protocol failure, if regular event inter-arrival times are actually expected. Consider for example the event of sending control information as background traffic. If such burst behavior occurs, control information might severely interfere with actual data transmission or even make data transmission during this burst impossible.

The other visual timing error indicator is event synchronization. This refers to a regular event that starts to occur on different nodes at about the same time. Refer to node 1 and 2 in Fig. 6. At about time 1200 both nodes start to enter the event handler almost simultaneously. Such behavior might be an error, if event scheduling on different nodes is expected to be a random process. For example regular beaconing intervals should not be in sync with other nodes in order to avoid too many beacon message collisions. Synchronization might occur, if inclusion of some random component was forgotten in the code or if using an error-prone random number generator for the beacon interval computation.

B. State Tracing

While event tracing aims on a fine grained inspection of a single event, state tracing provides a coarser view on protocol behavior and interaction. The state tracing concept aggregates event handler calls into specific classes. Again for every node a timeline is visualized. The timeline contains bars which extend over the time event handlers from a certain class are called. As soon an event handler from a different class is called, the bar will change. For producing such trace, the evaluator requires the same input as for event tracing. In addition a classification of event handler calls needs to be provided. Such classification declares the classes itself and assigns certain event handler calls to those classes. Any event handler call that is not in this class will be ignored in the evaluation. As with event tracing the result is an PDF file, which contains one or more lines of time line representations. Refer to Fig. 7 for an example of a single line state tracing result.

![State Tracing Example](image)

Fig. 7. A state tracing example.

Finding the erroneous code when an error has been found for a certain node at a certain time is one helpful application of this diagram. Consider for example, that a state machine implementation is to be inspected. Say every possible state implements its own handlers for the considered events. Then a reasonable classification will be aggregating the handlers of every state into a different class. Consider now that for the example depicted in Fig. 7 we figured out by some other means that an error occurred on node 3 at time 1200. By inspecting the state trace we know that at that time node was calling event handlers belonging to the dark grey state. Thus, we can narrow our failure search to the event handlers aggregated into the dark grey state.

The diagram can also be useful for detecting the occurrence of state oscillation. We denote with this term an unusual frequent change of protocol states on a node. Consider the
example depicted in Fig. 7. Suppose the colors refer to the states of a clustering protocol. Light grey means undecided, grey means cluster member, and dark grey means cluster head. At 1000 all nodes are in state undecided. Then cluster head and member roles are assigned to the nodes. Around time 1100 roles of 3 and 4 interchange which may be ok if reclustering is an allowed feature. However, at around 1200 nodes 1 and 2 start to exhibit an suspicious behavior with their states starting to change between cluster head and member very frequently. This may be an indication of a failure in the clustering protocols reclustering strategy.

C. Accumulation Diagrams

This visualization feature comprises two classes, which aim at a pictorial representation of network-wide effects at large scale. The first one, referred as path accumulation diagram [2], depicts the frequency of links taken by the messages originating from a certain node and belonging to a certain message class (see Fig. 8a and 8b). In our current implementation, the evaluator requires the source node, a single message log file, the type of the messages to be presented, a time frame the figure should be created for, and a mapping of nodes to physical locations in the network.

For each link the number of messages passing that link within a certain time interval is counted. The frequency is then depicted with different line thicknesses, i.e., the most frequently used links are presented with the thickest line. This simple representation is an effective way to get a first impression about the global behavior of a system under test. For instance, routing messages might disperse over the network like depicted in Fig. 8a, disseminating the forwarding burden over the network, but resulting in longer path deviations on the other hand. The opposite behavior, i.e., a protocol that concentrates the path on a few selected next hop nodes, might show a figure like sketched in Fig. 8b. By just visually comparing the diagrams resulting from different protocols, one also gets a first impression about how these protocols may compare in terms of hop count. In the example, Fig. 8a suggests a higher hop count when compared to Fig. 8b. This may also give a first indication on delay characteristics which, however, needs to be investigated separately, if significant evidence is required.

The same visualization principle can also be applied to nodes instead of links in order to show load distribution among network nodes in an intuitive manner. We term this a load accumulation diagram. Different node thicknesses are used to visualize load placed on the nodes according to a certain metric. Fig. 8c and 8d show an example load accumulation diagram. Intuitively, Fig. 8c suggests a more balanced load than Fig. 8d, in which a concentration of the load appears in the network center. Assume, e.g., the figures depict the total energy consumption per node during the whole measurement. When aiming at a protocol design that tries to maximize network lifetime by balancing energy load among all nodes, then both diagrams suggest favoring the protocol that produced the left-hand outcome.

In our implementation the evaluator requires the physical node locations, the evaluation duration, and the evaluation metric. They have to be provided via the context file. Example metrics we considered in this work include the number of message transmissions and receptions, the number of calls of a certain function, and the sum over the nodes sojourn time within a certain state. Depending on the investigated metric, either a single message log file or a call-chain log file has to be provided to the evaluator.

D. Statistics

The concepts presented so far focus on extracting visual data from automatically generated log files. We believe that this is a valuable feature to capture intentioned or failure behavior of a protocol in an intuitive manner. In fact, however, the generated log files offer more than that. Once a significant amount of data has been produced, a rigorous statistical analysis can be run over the data set. We list some common example statistical values that can automatically be extracted from the log files created by instrumented code.

The most common statistical values can be inferred by counting rows or by considering the values of rows in the call chain and message log files matching a certain criterion. One examples would be the fraction of transmissions/receptions of messages of a certain type over the total number of message transmissions. Another one would be using entry time, exit time, and handler ID to infer the number of times a node did enter/exit, and the time fraction a node spent in a certain event handler or in a group of event handlers.

Moreover, we can infer simple statistical data on the spatial distribution of certain message types. Using the message log files’ node, origin, and seqno columns, for each individual message we can infer the hop count distance the message travels from its creation until it disappears. Therefore, we are able to compute the expected hop count for each message type. Message types used locally, e.g., for neighbor discovery,
can thus be distinguished automatically from messages used globally, e.g., flooding messages used for route discovery.

We also implemented statistical data extraction about routing paths, which in contrast to the above mentioned requires a bit more than automatically generated log files. Extracting statistics about routing paths require additional context information about the message destination. This is available due to manually inserted consume code extensions during the instrumentation process. Such instrumentation will in turn result in message log files that contain information about message reception at the routing destination. An alternative way is to leave the code untouched and provide context information about source and destination nodes during the evaluation pass. Using the message log file’s time, origin, and seqno columns, in both cases we are able to infer the traditional routing metrics like average hop count, average delay, and success rate.

V. CONCEPT EVALUATION

The concepts introduced so far are evaluated by instrumenting an example application and three routing protocols. In all cases TOSSIM is used for simulating the instrumented code. Here, the process of logging data is simplified by the fact that the information can be logged directly into files. The configuration of the instrumenter for TOSSIM is as follows. We have provided the modules responsible for creating, sending, and receiving packets and code templates producing tracing information as described in Sect. III-B. An example code snippet for tracing packet reception is shown in Listing 1.

In the following a detailed analysis of TinyAID is given. The evaluation of the SelfWISE framework is described in Sect. V-A. The instrumentation for tracing the packet flow in routing protocols is discussed in Sect. V-B and V-C. The overhead introduced by TinyAID when using TOSSIM is benchmarked in Sect. V-D.

tossim_header_t * header = getHeader(msg);
dbg_clear("TINYAID_PACKET_TRACING",
"%d%d%d%d%d%n",

Listing 1. Code template for packet reception

Using TinyAID on real hardware in a deployed network requires more advanced mechanisms to collect the logged data. Storing the information on the nodes limits the runtime of the experiment due to the limited memory. The best choice is to have a wired backbone network that can be used for logging purposes. The overhead introduced by logging the data via the serial line is to large (around 2 ms) for most situations. Therefore, a special hardware solution is needed in order to extract the data and collect it at a central point. This additional hardware is part of our future work. In this paper we are concentrating on the principal concept of automated instrumentation and its usability in the area of wireless sensor networks.

A. Use Case: SelfWISE

SelfWISE is a framework for evaluating self-stabilizing algorithms developed at the Institute of Telematics at Hamburg University of Technology [3]. This software contains around 9,000 lines of nesC code. TinyAID is used to inspect the behavior of the SelfWISE framework.

The execution of the self-stabilizing algorithms is based on rounds that must be synchronized and a shared node state that is accomplished by periodical broadcasts. The broadcasts are scheduled at a random point in time to reduce the number of collisions. To evaluate the quality of the implementation, the event inspection of TinyAID is used. Without any knowledge of the internal structure, TinyAID creates instrumentation points in 101 different functions. To investigate the synchronization of the round timer and the broadcast, two function calls where selected: SyncTimer.fired and Paket.send. The resulting event trace is shown in Fig. 9.

Fig. 9. Event tracing of SelfWISE

This evaluation shows that the synchronization of the round timer works perfectly and that the node state is broadcasted at a random point during the middle of each round. The cyclic patterns of the broadcast were not expected. After further investigations we found out that the implementation of the RandomMlcgC component produces insufficient random numbers when using TOSSIM. In real deployments events like packet reception and code runtime introduce a different runtime behavior, so that this effect is not dominant. In TOSSIM, however, the random numbers create these periodical patterns, which leads to an abnormal behavior over time with reoccurring collisions between the same nodes. Changing the implementation of the RandomMlcgC components in such a way that it uses the rand function of the libc in TOSSIM creates a more randomly distributed message broadcast. Without the automated instrumentation and evaluation
support of TinyAID we may have never become aware of this effect.

The next example evaluation shows the application of state tracing. A self-stabilizing algorithm is executed whenever a node sees an irregularity within its neighborhood. To avoid gratuitous executions the number of nodes that execute the algorithm is reduced by a Bernoulli trial. The behavior over time is shown in Fig. 10. The algorithm execution starts at time 53 and stabilizes at 64. The dark rectangle visualizes that in this round a node has executed the algorithm (done by calling RuleEngine.abort) and the gray rectangle indicates that the node wants to execute, but the Bernoulli trial was negative (RuleEngine.execute).

![Fig. 10. State tracing of SelfWISE](image)

B. Use Case: Routing Protocols

In this section, we will demonstrate and discuss how our concept of automated packet tracing can be integrated into already implemented protocols. Therefore, three different routing protocols to carry out our analysis on automated packet tracing and the ease of its implementation have been chosen.

The first one is TYMO, an implementation of the well-known DYMO protocol [4], which is included in the TinyOS 2.x code base. TYMO uses internal message types for route requests and route replies. These are sent in order to query and establish a new route, if a forwarding node does not know where to send a given message. The second routing protocol is Dynamic Source Routing (DSR) [5], which follows a similar concept. The third protocol considered is Greedy Routing [6]. Messages are forwarded using the positions of forwarding nodes and the destination. The greedy aspect is realized by each forwarder considering only neighbor nodes closer to the destination and sending the message to the neighbor closest to the destination. The implementations we used for DSR and Greedy Routing originate from [2].

As noted in Sect. III-B, tracing packet sending and receiving is automatically added to the compiled code, if Active Messaging is used. However, Packet.clear must be called at the appropriate places in order to make packet tracing work. The following, general issues must be considered first. If the routing protocol under investigation makes use of internal packet types, it must be assured that the corresponding calls to Packet.clear are performed, whenever a new internal packet is created. Secondly, routing messages, whether originated on the same node or received and forwarded, must be stored as a complete TinyOS message in a buffer. This is necessary in order to keep message IDs, i.e., packet origin and sequence number, intact.

Calling Packet.clear for data messages, that are passed from the application to the routing layer, must be done at the application layer for two reasons. Firstly, following the Active Messaging concept, packet clearing should be performed on creation of a new packet. Clearly, data packets are created at the application layer. Secondly, accurate timing information about packet creation can only be guaranteed, if Packet.clear is called upon actual packet creation.

In order to prepare the three routing protocols for automated packet tracing, few changes and additions had to be applied. Neither the implementation of DSR nor Greedy did use the concept of Packet.clear. Hence, the corresponding calls had to be added for all internal packet types. Furthermore, both implementations only copied the data part of forwarded data messages into the buffer, although the latter consists of complete TinyOS messages. A minor change in both protocols expunged this problem. These simple and quickly applied changes enabled the two protocol implementations to support automated packet tracing. TYMO does not make use of Packet.clear either, so that the appropriate calls had to be added. Moreover, it knows but two packet types. One is intended for internal message exchange, i.e., sending protocol data, whereas the other one is used for routing messages. The former uses subtypes that are stored in the actual packet payload to allow for different protocol message subtypes. As a result, it is required to edit comparably many lines to make the different subtypes visible in the trace log. For a first analysis of our tools, we decided not to take this step and abdicate tracing of the different subtypes.

Besides preparing the routing protocols, we added the required calls of Packet.clear upon packet creation to our test application. We also added consume upon reception of data messages on the destination node.

C. Performance Metrics

In order to illustrate the power of packet tracing, we will show and discuss visual and statistical analyses obtained from automated traces. We have used the routing protocols introduced in the previous section to show the benefit of packet tracing. The figures and tables in this section have been created with analysis tools that take origin, destination and inspected types as parameters.

All results are based on the same topology of 25 nodes, where the same two nodes serve as data origin and data sink. 10 data packets are created by the origin with a period of 2 seconds. One simulation has been run for each of the three routing protocols. At this point, it is not our intention
to actually compare the three routing protocols. We plainly are concerned about outlining illustrative examples of utilizing packet tracing for protocol analysis and comparison.

Figure 11 visualizes packet flow and an energy-consumption equivalent. Packet flow, as depicted in the upper row, shows the number of packets intentionally sent on each link, i.e., not counting snooped (overheard) packets. The number of packets sent between two nodes are used as a linear scaling factor for the displayed line width of the corresponding edge. From the visualization it is apparent that TYMO and DSR cause packet flow in the whole network in order to establish routes. In opposition to this, Greedy (Fig. 11) causes local packet flow only, because routing decisions are based upon the position of the destination and a forwarder’s neighbors. Packet flow between origin (lower left) and sink (upper right) is considerably higher than in other regions of the network. Furthermore, the share of packet flow caused by routing packets can be compared to that caused by internal routing protocol messages on the actual data paths (gray versus magenta edge widths).

The mid row of Fig. 11 depicts the number of sent and received messages per node. This metric can be used as an energy-consumption equivalent. All figures reveal that energy
consumption is higher the closer a node is to the actual data path. This is caused by packet overhearing and internal protocol message exchange. In opposition to packet flow observed for Greedy Routing, nodes far apart from the data path still consume a small amount of energy for overhearing routing packets.

Another interesting aspect of routing protocols and sensor network applications is packet type distribution. Here, it is investigated which nodes in the network are sending or receiving certain packet types. The lower row in Fig. 11 displays packet types handled by all nodes in our example topology. In case of DSR, e.g., only nodes on the data path send and receive packet types 10 (route reply) and 12 (data packet). In contrast, all nodes either send or receive packet type 9, which corresponds to route requests. Note that packet reception here implies that a node receives a packet destined to itself or to the broadcast address.

Besides visual analysis, tables with descriptive statistics can also be derived from the generated message traces. Example data is shown in Table I. The number of packets sent reveals major differences between the protocols. Another comparison depicted in the table is the latency between data packet creation and their reception at the data sink. Here, it shows that TYMO produces the lowest latency. The high maximum latency of Greedy Routing is due to the fact that the implementation we used from [2] is a reactive one. In the absence of traffic, no neighborhood information is maintained. Whenever a node receives a packet, it produces a neighbor request. The node then gathers information about neighbor node positions in order to make the correct next routing decision. As pointed out in [2], this technique is a delay versus reliability tradeoff.

In conclusion, packet tracing enables protocol designers to gain an in-depth look at key metrics of their protocols. It additionally allows for easy comparison between protocols, whether using visual representation or plain figures – whichever seems more convenient or appropriate.

### D. Overhead

The results in Table II are showing the overhead introduced of TinyAID. The simulations are based on the SelfWISE framework simulating 100 seconds, which means 100 rounds with one broadcast per round and node. The simulations are run on topologies from 4 up to 100 nodes. The nodes are arranged in a grid in such a way that a node in the center has exactly four neighbors. The simulations are performed with the original SelfWISE framework and three different instrumented versions. In the selective version exactly two functions are instrumented, namely those that are used to evaluate the results shown in Fig. 9. The partially instrumented code monitors all functions that are not part of the TinyOS operating system. In the complete instrumented version every function is instrumented. The overhead of simulation time is relative to the simulation time without instrumentation. The number of events represents the number of entering and leaving events. The results reveal that instrumentation must performed carefully. Otherwise the simulation time will increased up to three times. If only selective functions are instrumented, the introduced runtime overhead is not significant. The size of the resulting TOSSIM library depends on the number of instrumented functions. The selective instrumentation does not increase the size. The partially instrumentation increases the size from 552 kB by around 1% to 559 kB.

Complete instrumentation leads to a non-functional executable, since the debug messages are printed before the simulation is setup. So at least 

```
/opt/tinyos-2.x/tos/lib/tossim
```

must be excluded in order to simulate the instrumented code.

In our simulations of the routing protocols, sending 10 data packets from one corner of a 25-node network to the opposite one produced trace files with sizes between 10 and 20 kB. This amount could fairly be reduced by a more compact way of trace file layout. However, estimating log file size is difficult, because the number of packets sent in a network is the major driver here. It depends on the number of nodes; protocols used for medium access, routing, etc.; and the application.

Packet tracing causes a two-fold overhead. Firstly, additional code must be executed in order to log data. In the case of simulation, this overhead only concerns simulation execution time. In a real testbed, however, the processor of a node must execute the tracing code, which may, e.g., change timings or energy consumption. Secondly, packet size is increased by currently 4 Bytes. Depending on the data payload size, this increase may become significant and cause side-effects, such as increased packet transmission times or packet loss rates.

### VI. RELATED WORK

The EvAnT framework [7] and the Rupeas language [8] are tools for sensor network analysis. Both approaches interpret a log file as an event collection with each row being an individual event. Each event is then specified by the column values. EvAnT and Rupeas feature event set processing, event set queries, and assertions for testing. The creation of log files, however, is not part of EvAnT and Rupeas. The system
under consideration has to be instrumented manually with log file generating code first. In contrast, our approach supports automated instrumentation for creating log files. By providing the instrumenter with the right code templates, TinyAID is able to create any specific log file format, in particular those which might then be used for EvAnT and Rupaeas.

The Sympathy [9] and Memento [10] network monitoring systems, and the concept of passive inspection [11] are focused on the message communication part of the system. In Memento and Sympathy the system under consideration is extended by additional code, which performs failure detection based on message monitoring. The same sensor network is then used to report logging data to a specific collecting node. Passive inspection of sensor networks follows a complimentary approach to Sympathy and Memento. Message log files are created by an additional deployment support network where every node owns two wireless transceivers. One transceiver is used in order to overhear all wireless sensor network traffic in the surroundings. The second transceiver, being a robust and high-bandwidth one, is used to transmit the results to a specific collecting node.

The presented TinyAID differs from Sympathy, Memento, and passive inspection. In these approaches system inspection always has to follow a black box approach, i.e., information about traffic patterns is used to infer information about code correctness on nodes. Moreover, there are no guarantees that all message communication failures are detected. In our approach code is instrumented directly at the points of message transmission. In addition, when an irregular behavior is observed, call-chain logging potentially supports finding the faulty module handlers directly.

Instrumentation approaches for wireless sensor networks are going beyond inspection of messages: EnviroLog [12] and Declarative Tracepoints [13]. In EnviroLog code has to be annotated manually first and then passed through a preprocessor before the final compilation pass. This preprocessing approach is comparable with the one presented by the TinyAID instrumenter. However, in contrast to EnviroLog, TinyAID exempts the programmer in many cases from touching the inspected code directly.

In the Declarative Tracepoint approach the high level declarative programming language TraceSQL for code instrumentation is introduced. Using this instrumentation language, the user is not required to manually touch any line of code under consideration. The language allows entering so-called action-associated check points into the source code. If the check point is passed and the check point predicate is satisfied, a certain action is performed. From the perspective of the language features, this approach appears to us the most general wireless sensor network instrumentation support. However, in contrast to the solution presented with TinyAID, the programmer has no control over what exactly is inserted into the instrumented code. It depends on the implementation of TraceSQL. In TinyAID code templates make the inserted code explicit and give the programmer full flexibility to tailor the templates to his needs. For example, the way logging is performed in a TOSSIM simulation may simply be done by using the dbg function, while logging in a real testbed deployment may be achieved by another code template for writing logging information to the serial port directly.

Support for message-flow tracing is another feature that distinguishes TinyAID from TraceSQL. While in TraceSQL check points may be added to the send/receive handlers of the Active Messaging module, there is no way to tag messages with an additional unique message identifier. In our approach unique message identification supports tracing message instances from the creating node to the nodes where the message is either dropped or delivered.

Other useful debugging approaches presented in the literature are NodeMD [14] and SNMS [15]. The emphasis of both approaches is a fine-grained node-level inspection of failure behavior. NodeMD focuses on specific node failures covering stack overflow, and lifelock/deadlock situations in multi-threaded environments. The goal is to catch such failures and provide the user with diagnostic information, which can then be used for troubleshooting, before the node becomes completely unusable. In SNMS the focus is on attribute export tagging variables, the tool provides a query-based health data collection and persistent event logging system. Fine-grained node-level debugging versus inspecting system-wide behavior is the main difference between these approaches and the TinyAID approach presented in this work.

| Number of Nodes | Without 3,04 | Selective 4508578 | Partially 9,53 | Complete 5.62 | Events 1.01 | 2819526 | 29.33 | 2.83 | 15132452 | 0,01 | 4.04 | 605924 | 22.31 | 11864168 | 2,82 | 998476 | 0,70 | 0.65 | 16.07 | 11.60 |
|-----------------|-------------|-------------------|--------------|--------------|------------|----------|-------|------|------------|------|-----|-----------|------|-----------|------|------|------|-------|
| 4               | 0.45        | 0.46              | 0.03         | 1344         | 0.65       | 105866   | 1.58  | 0.57 | 303928    | 3.74 | 2.72 | 1506068   | 2,54 | 613372   |
| 9               | 1.01        | 1.01              | 0.01         | 3024         | 1.58       | 605924   | 6.85  | 2.82 | 2793508   | 0.01 | 4.04 | 605924    | 22.31 | 11864168 |
| 16              | 1.79        | 1.82              | 0.02         | 5376         | 2.94       | 998476   | 11.04 | 2.96 | 4465828   | 0,01 | 6512520 |
| 25              | 2.79        | 2.83              | 0.02         | 8400         | 4.71       | 1502018  | 16.15 | 3.00 | 6512520   | 2,54 | 613372 |
| 36              | 4.04        | 4.13              | 0.02         | 12096        | 6.89       | 2102180  | 22.31 | 3.04 | 8992210   | 0,01 | 6512520 |
| 49              | 5.52        | 5.62              | 0.02         | 16464        | 9.53       | 2819526  | 29.33 | 3.04 | 11864168  | 0,01 | 6512520 |
| 64              | 7.26        | 7.36              | 0.01         | 21504        | 12.70      | 3608232  | 37.07 | 3.04 | 15132452  | 0,01 | 6512520 |
| 81              | 9.18        | 9.28              | 0.01         | 27216        | 16.07      | 4508578  | 46.45 | 3.09 | 18802426  | 0,01 | 6512520 |
| 100             | 11.36       | 11.60             | 0.02         | 33600        | 19.93      | 37.07    | 8992210 | 3608232 |

**TABLE II**

**EVALUATION OF THE OVERHEAD INTRODUCED BY TINYAID**
In this work we devised a major part on automated evaluation with visualization features that enables the developer to capture complex network-wide effects in a visual, intuitive manner. To the best of our knowledge little effort has been spent in that direction so far. As an exception, the routing path visualization concept presented in the Rupeas publication [8] is close to the spirit of what we mean with capturing network-wide effects in an intuitive way. Compared to that visualization concept, we see the concepts presented in this work as a complement and an extension of a hopefully more and more growing set of available visualization concepts.

VII. CONCLUSION

In this paper a code instrumentation and evaluation tool for the TinyOS community is presented. We highlighted the advantages of automatic instrumentation support over manual instrumentation and presented a simple but effective way for automated code instrumentation. The so-instrumented code produces log information that covers two aspects: logs of the call chain and message flow. The automatic evaluation tool based on such logged data, enables a programmer to capture system-wide behavior in a visual and intuitive manner. In addition, statistical performance quantities can immediately be extracted from the generated log files in an uniform way.

The empirical studies performed with TinyAID show that automated instrumentation and evaluation are a valuable support for TinyOS-based programming. For instance, a synchronization problem in the SelFWISE simulation was figured out by just looking at the visual representation of an event trace. In addition, a set of routing protocols originating from different programmers are compared. Spending only little effort on reading the code, the protocols are evaluated, both visually and also by means of some plain statistical data.

The main empirical studies in this paper were performed by simulation only. The next development step is to add support for real sensor network deployments. It is to be considered how the logged data can be gathered in an effective way and how to order events system wide when node clocks are not synchronized. In our future work plan we envision a combined hard- and software approach that is currently in its initial design phase, based on the presented conceptual findings in this paper.

An additional, future project we see is the integration of other concepts into TinyAID, e.g., by supporting the appropriate log file structure TinyAID may be combined with Rupeas. Moreover, a future extension of TinyAID may be the use of the TraceSQL language concept. So far, TraceSQL did not leave us enough flexibility on which code is added as trace points; a reason why we chose the code template approach. In a future extension, however, TraceSQL and the concept of code templates might be combined.

ACKNOWLEDGMENT

The authors would like to thank René Steinrücken for his help during the implementation of the TinyAID instrumentation software.

REFERENCES

DCOSS Workshop:

Third International Workshop on Information Theory for Sensor Networks (WITS)
Selected CRC Polynomials Can Correct Errors and Thus Reduce Retransmission
Travis Mandel, Jens Mache

Abstract- For wireless sensor networks, minimizing communication is crucial to improve energy consumption and thus lifetime. Whereas the standard way to deal with transmission errors is retransmission (automatic repeat request ARQ), in this paper we investigate an alternative: correcting bit errors using Cyclic Redundancy Checks CRCs (which are already used for error detection). Selected CRC polynomials - including CCITT-16 which is used by IEEE 802.15.4 and TinyOS - can correct 1-bit errors in up to 240 bits of data. We present our send-check-confirm (SCC) protocol that reduces retransmission without sacrificing reliability since corrections are validated. In addition, we list 64 16-bit candidate CRC polynomials that can correct for 1- and 2-bit errors in less than 240 bits of data.

II. BACKGROUND AND RELATED WORK

A. Cyclic Redundancy Checks
Cyclic redundancy checks use binary polynomial division to detect errors. To compute a CRC one must first pick a generator polynomial G. In practice, there are certain recommended choices of G that increase the error detecting ability of our CRC algorithm. [16] As an example, let G = (x^3+1). We represent the coefficients of the polynomial as bits, so G=1001. We also know that the first bit of G will always be a 1, so we only need to store n=3 bits, 001. Now assume our original data M = 101110. To compute the checksum of M, we must first append n bits to the end to create D=101110000 and take CRC_G(D). This is simply binary polynomial division of G into D.

\[
\begin{array}{c}
  101011 \\
  1001 | 101110000 \\
  \hline
  1001 \\
  101 \\
  000 \\
  1010 \\
  1001 \\
  110 \\
  000 \\
  1100 \\
  1001 \\
  1010 \\
  1001 \\
  011
\end{array}
\]

So the CRC_G(101110000) is 011 in binary (3 in decimal). Then our transmitted message M' consists of M concatenated with its CRC, or 1011100001001. The receiver takes the CRC of what it receives. If there is no corruption, this CRC will be zero, otherwise, an error is detected. In our example, we take CRC_G (101110011)=0, but if the last bit is corrupted CRC_G (101110010)=1.

B. Related Work
Relatively little work has been done in using CRCs for the correction of bit errors.

Much of the previous work has been concerned with only a single generator polynomial , such as a paper titled Single Bit Error Correction Implementation in CRC-16 on FPGA [5], which offers optimizations for single-bit correction that are only applicable for CRC-16-X25. They mention that their results could be easily modified for another 16-bit...
polynomial, but their designer relies heavily on the fact that the generator polynomial has exactly 16 bits.

[4] discusses further hardware-level optimizations for the table used in single-bit error correction (this table is described in section III). They claim their methods work for any received message length and any formal CRC generator polynomial, but do not offer a thorough discussion of single-bit error correction over generalized polynomials and messages.

McDaniel’s paper on single-bit error correction [1] gives a much more thorough analysis and is discussed in detail in section III.

Related work regarding reduced retransmission includes [10, 11, 9, 8] and is discussed further in Section VII.

III. SINGLE BIT ERROR CORRECTION

In [1] McDaniel describes a tabular method for correcting single bit errors given a n+1-bit generator polynomial G. The first step is to precompute an error correction table T for the particular choice of G. To do this, one creates a message Z of length \(2^n-1\) that is composed entirely of zeros. Then one changes each consecutive bit i of Z to 1, and fills table T such that \(T[\text{CRC}_i(z)] = i\). For example, n=3 and \(G = x^3 + x + 1\) yields \(z_3=1000000\) and \(\text{CRC}_i(z_3)=5, z_2=0100000\) and \(\text{CRC}_i(z_2)=7, \text{CRC}_i(z_2)=6, \text{CRC}_i(z_2)=3, \text{CRC}_i(z_2)=4, \text{CRC}_i(z_2)=2, \text{CRC}_i(z_2)=1\). Thus T=[correct, 7, 6, 4, 5, 1, 3, 2]. Note that the zeroth entry in the table will not be used since a CRC of zero indicates correctness, thus there is no bit to correct. If no two mutations of Z have the same CRC, meaning \(\text{CRC}_i\) is injective for every \(z_i\), then G is a suitable polynomial for error correction. This means given any message \(M'\) (consisting of an original message M of size \(L=2^n-1\) with an n-bit checksum C appended), one can correct any single bit error.

To correct single-bit errors, one takes the \(\text{CRC}_i(M')=C\). If \(C_2\) is zero, the initial message is correct. Otherwise, \(T[C_2]\) is the index at which the single bit error occurred. Continuing the above example, \(M=1100\) yields \(C_2=\text{CRC}_i(1100)=010\) and \(M'=M\) concatenated with \(C_1=1100000\). Incorrect reception of 1101010 yields \(C_f=\text{CRC}_i(1101010)=3\). Since this CRC is not zero, we examine our table. \(T[3]=4\), which indicates that bit 4 was corrupted. Thus flipping bit four gives us 1100001, the correct message.

A. Generator Polynomials

McDaniel’s paper [1] claimed that no commonly used generator polynomials could be used for error correction. This would present some obstacles to the widespread use of error correcting CRCs, as there are certain error detecting properties of most commonly-used generator polynomials [16] that make them more resilient than others to common forms of corruption, such as burst errors. However, McDaniel’s claim is untrue. We have verified that the CRC-8-CCITT \((x^8 + x^7 + x^6 + x^5 + 1)\) standard does indeed have this error-correcting property. Other polynomials also have this capability with smaller message sizes, see section III.B.2, below.

B. Dealing with Different Message Sizes

McDaniel [1] also claimed that the table T (described above) can handle any message size of \(\leq 2^n-1\). This is not the case. Continuing the above example, \(M=11\) yields \(C_f=\text{CRC}_i(11)=101\) and \(M'=M\) concatenated with \(C_1=11101\). Incorrect reception of 01101 yields \(C_f=\text{CRC}_i(01101)=6\). Since this CRC is not zero, we examine our table. \(T[6]=3\), which indicates that bit 3 was corrupted. But flipping bit 3 gives us 01001 which is incorrect.

McDaniel’s original method can only handle messages of exactly \(2^n-1\) n-bits. Thus messages using 16 bit error-correcting CRCs must contain \(2^{16}-16 = 8\) KiloBytes of data (whereas TinyOS [7] allows at most 29 data Bytes). Being forced to send messages of exactly this length across the medium would require an unreasonable amount of bandwidth and energy, as increasing the amount of data in each packet requires more of these resources. Fixing message length in this way would also increase the probability that several bits of our message were corrupted. We have devised two solutions to this problem:

1) "Bit “Padding”

By padding the message only while calculating the CRC, not transmitting the padded bits, we can avoid such problems. When calculating \(C_f\), we simply iterate \(2^n-1\) times through the bits of the message, considering the bit to be zero if we have run off the end of the message. Then when calculating \(C_f\), we iterate through all but the last n bits of the message, add zeros until we reach \(2^n-1\), and then iterate through the last n message bits. For example, if \(n=3\) and \(M'=11101\), we take \(\text{CRC}_i(1101011)\). In such a manner we still have to send only \(L+n\) bits across the medium, using a minimum of bandwidth, but can accommodate any message of length \(L=2^n-1\).

2) Initialization

If we know a maximum data length \(X<2^n-1\), we can make \(Z\) of length \(X+n\) when we calculating the correction table. Then we only have to bit pad smaller messages to length \(X\). In addition to potentially saving bandwidth, this also saves time, as the CRC method has to consider fewer padding bits. For example, \(n=3\), \(G=x^3 + x + 1\) and \(X=2\) yields \(\text{CRC}_i(00001)=6, \text{CRC}_i(00010)=3, \text{CRC}_i(00100)=4, \text{CRC}_i(01000)=2, \text{CRC}_i(10000)=1\), thus \(T=[\text{correct, 5, 4, 2, 3, }-1, \rightarrow]\), and we can correct any message up to length \(X=2\). Given a maximum data length \(X=2^n-1\), the table is not completely filled and more generator polynomials are suitable for error correction.

IV. PROTOCOLS

The major problem with replacing the original automatic repeat request (ARQ) protocol with a single-bit send-correct (SC) protocol is that it will almost certainly lead to wrong corrections. To see this, assume the message \(M'\) has two errors. Then when \(M'\) is received, \(C_f\) will be computed and found to be non-zero. Thus \(C_2\) will be looked up in the table, indicating a bad bit \(R\) (assuming the table has a mapping for that CRC). When the checksum C is taken of \(M'\) with \(R\) flipped, \(C\) will be zero, since the table was constructed to have that property. This is unacceptable as now we perceive a message as correct that is in fact incorrect: we started with two errors and corrected only one bit. Thus one can only use the SC protocol if one can be
certain that all messages have either 0 or 1-bit errors, and no multiple-bit errors.

Fig. 1. ARQ and SC protocols

A. Proposed SCC protocol

In order to resolve this issue, we need some sort of confirmation of the bit that we suspect of being faulty. We propose a single-bit send-check-confirm (SCC) protocol which includes this safeguard. The first step after hypothesizing an error bit R is to check whether or not R falls within the padding area, as if it is, it cannot possibly be right. Otherwise we send to the original sender R’s index concatenated with R’s value. Because this message is much shorter the original (~log \( n \) bits), the probability of corruption is much lower. To guard against corruption of this short message, one can either append a CRC for error detection only, or use some simple error correction scheme such as triple modular redundancy. When the original sender receives the message, they examine the original data. If the received bit is indeed correct, the sender sends a very short message indicating this fact. The mechanism is very similar to the HTTP protocol’s “304 Not Modified” message, which is a short message sent in place of a resend of a web document if there is no need to resend the object (i.e. the document has not been modified since it was last accessed) [3].

If the bit is not correct, the sender resends the original message. Thus, if there is a correctable error, we have dramatically shortened the message size, and if not, the resend penalty remains the same as in the original ARQ protocol. Fig. 1 depicts the ARQ and SC protocols, and Fig. 2 depicts our proposed SCC protocol.

Fig. 2. Proposed SCC protocol, in the case of a single bit error. Note that message size is severely decreased, as \( M' \) does not have to be sent a second time.

We implemented the above algorithms, see class diagram in Fig. 3.

Fig. 3. Class Diagram

V. PERFORMANCE ANALYSIS:

There is a potential for error in our SCC protocol, in that it is possible to have the server reply that we have corrected all errors when in fact we have not. We call this “undercorrecting” because some errors remain after the initial correction. We will now derive the probability that such an event occurs.

We can model bit errors as a random process in which with some probability \( p \) we flip each bit in our data. [14] This is equivalent to \( X=\text{Binomial}(n,p) \), where \( n \) is the number of corruptible bits. In our case, there are \( n=248 \) bits (The 232 message bits plus 16 CRC bits) . \( E[x]=np \) is between 1 and 1.5 since [2] tells us that around 50% of bit errors are single-bit errors. We would like to know the probability that the server responds that our guess is correct(event A). This is the probability that the table has an entry and the bit in the table is one of the bits that was flipped originally. We do not care about the case where \( X=0 \), since we are assuming for argument’s sake an infallible CRC. If \( X \) is 1, we know that the table has an entry (because of how it is constructed) and that entry is correct (since we have only a 1-bit error). Happen, if \( x \) is greater than 1, we know that the probability that the table is full is independent of the probability that the bit chosen is correct. The probability the table entry chosen has a value is the number of entries in the table over the table size (we have assumed a uniform distribution of errors). The probability the bit is flipped is the number of bits chosen, \( k \), over the total number of possible bits, 248. Hence:
Pr[A] = Pr[x = 1] + \sum_{k=2}^{248} \frac{248}{2^{16}} \times Pr[x = k] \times k \times \frac{1}{2^{48}}

=Pr[x = 1] + \frac{1}{2^{16}} \sum_{k=2}^{248} k Pr[x = k]

=Pr[x = 1] + \frac{1}{2^{16}} (E[k] - 1 Pr[x = 1] - 0 Pr[x = 0])

=Pr[x = 1] + \frac{1}{2^{16}} (np - Pr[x = 1])

Notice that either x is one and we have corrected our data correctly, or we have undercorrected our data. The probability of under-correcting our data is \( \frac{1}{2^{48}} \), which is around \( \frac{1}{2^{16}} \) since np is about 1.5 and Pr[x=1] is around .5 . Thus the probability of under-correcting our data is only \( \frac{1}{2^{48}} \) with our protocol, which is acceptably small compared to the probability that the CRC fails to detect the error in the first place.

VI. MULTIPLE-BIT ERROR CORRECTION

A. Motivation

As shown in Fig. 4, TinyOS [7] uses a message format that allows at most 29 data Bytes (232 data bits) but 16 CRC bits. We found that the generator polynomial used in the IEEE 802.15.4 standard, \( x^{16} + x^2 + x + 1 \), is usable for single-bit-error correction over this message length. However an 8-bit CRC would already be sufficient to correct single bit errors in up to \( 2^8-1=255 \) bits of data. For formats such as TinyOS’ which use a longer CRC, we can use the extra information provided by the CRC to correct multiple erroneous bits.

B. Implementation and Analysis

In order to perform multiple bit error correction, one needs a CRC of length n such that \( 2^n \) is much larger than the maximum message length L. In fact, to perform k-bit error correction, \( 2^n \gg \binom{L}{k} + \binom{L}{k+1} + \cdots + \binom{L}{L} \). The reason for this is that the CRCs of 1,2,…,k-bit corrupted z’s must be unique, so that our table mapping is injective. Once we have a table T with such a unique mapping, we may proceed with the SCC protocol, with the difference being that our table can indicate that several bits are corrupted, and we must send k locations and k values to be verified. Note that we will still receive only a confirmation or a resend, even if only one hypothesized error bit is incorrect.

We tested all 16-bit CRC polynomials such that all one and two bit errors in 232 bits of data mapped to unique CRC values. Out of the \( 2^{16} \) possibilities, we found 64 polynomials that can correct for one and two bit errors, see Table 1. Fig. 5 shows that as the maximum message (data) size increases, at first a very large number of possible polynomials exist, but this decreases quite abruptly after 113 bits of message (data) length, and remains fairly constant until 240 bits, at which point it is not possible to perform two bit error correction with a 16-bit CRC. The exact reason for such abrupt decreases in polynomial availability is left for future work. We also noticed that, in general, the polynomials that work for a given message length work for all lengths less than that, which explains why the graph is monotonically decreasing.

We also modified our protocol such that the receiver may detect 1 or 2 bit errors and send the locations and values to the original sender. This causes only a minor increase in communication, and the runtime is not impacted. However, the size of the correction table is now 128 KB, as compared with 0.25KB for single-bit correction’. The advantage of multiple-bit error correction is further reduced retransmissions.

<table>
<thead>
<tr>
<th>Table I. LIST OF 64 CRC16 GENERATOR POLYNOMIALS THAT CAN CORRECT 1- AND 2-BIT ERRORS IN 232 BITS OF DATA.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000111000000111</td>
</tr>
<tr>
<td>0001101010010101</td>
</tr>
<tr>
<td>0010001000010001</td>
</tr>
<tr>
<td>0010011011110011</td>
</tr>
<tr>
<td>0010100101101011</td>
</tr>
<tr>
<td>0011110110101011</td>
</tr>
<tr>
<td>0011110111111011</td>
</tr>
<tr>
<td>0011110111111011</td>
</tr>
<tr>
<td>0100011111111011</td>
</tr>
<tr>
<td>0101001011111111</td>
</tr>
<tr>
<td>0101010101101011</td>
</tr>
<tr>
<td>0110001011011011</td>
</tr>
<tr>
<td>0110001011010011</td>
</tr>
<tr>
<td>0111001011111111</td>
</tr>
<tr>
<td>0111001011110011</td>
</tr>
<tr>
<td>1000000100001001</td>
</tr>
<tr>
<td>1000000100000001</td>
</tr>
<tr>
<td>1000001110000111</td>
</tr>
<tr>
<td>1000001110000111</td>
</tr>
<tr>
<td>1000110011111111</td>
</tr>
<tr>
<td>1000110011110001</td>
</tr>
<tr>
<td>1001011100011101</td>
</tr>
<tr>
<td>1001011100010101</td>
</tr>
<tr>
<td>1001011100010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
<tr>
<td>1001110011010011</td>
</tr>
</tbody>
</table>

1 Here is how these numbers were computed: For single-bit correction, we have 232 data bits and 8 CRC bits, for a total of 240 bits. So there are 240 potential bad indices, and we can represent the index number with 8 bits. A CRC-8 correction table has \( 2^8 \) rows of a byte each =25KB. For double-bit correction, we have 232 data bits and 16 CRC bits, for a total of 248 bits. So there are 248 potential bad indices, and we can represent the index number with 8 bits. A CRC-16 correction table has \( 2^{16} \) rows of a byte each =64KB. But each entry in the table has two possible indices, so we have 128KB of space for the double-bit error correction table.
In number of generator polynomials that work for e.g. its 0-16-Its is that ed by IEEE 802.15.4 and TinyOS, can he corrupted packet and combines denced correct. Harq protocol, which employs orrection (HARQ) protocol, which employs error situations does not depend on a next packet. Our work i baseband level network coding is used to mix the probability retransmitted packet, thereby improving packet in decoding, avoids retransmissions that are to be caused by collisions and (2) our error errors do not have errors do not have errors. Our work is different since (1) our errors depend on a next packet and (2) our error-correcting CRC need less computational resources and energy. ZigZag decoding [8] avoids retransmissions - that are caused by collisions - by decoding the non-overlapping pieces. Our work is different since (1) our errors do not have to be caused by collisions and (2) our error-correcting CRC needs less computational resources and energy. Reed-Solomon Error Correction is a widely used correction scheme, however one of its faults is that it fails to efficiently correct or detect errors above a certain corruption level. [12] Thus many network architectures Hybrid ARQ (HARQ) protocol, which employs Reed-Solomon error correction in conjunction with a detecting CRC. [13] This allows the network to be able to correct small errors, but also detect high-error situations that Reed-Solomon cannot correct. However, our protocol uses only the CRC, thus avoiding the extra Reed-Solomon repetition needed by the HARQ detection-correction scheme.

VIII. CONCLUSION

Wireless communication faces transmission errors, but reducing retransmission can extend the lifetime of energy-constrained sensor networks. Our main contributions are summarized as follows:

- It is possible to correct errors using CRCs already present in sensor networks. Error correction can drastically reduce communication as resending information becomes necessary less frequently.
- We presented a simple and effective send-check-confirm (SCC) protocol that reduces retransmission, but reliability is not sacrificed since corrections are validated. The mechanism is very similar to the HTTP protocols’s “304 Not Modified” message (indicating that resending the document is not necessary).
- Many 16-bit CRC polynomials, including $x^{16} + x^{12} + x^5 + 1$ which is used by IEEE 802.15.4 and TinyOS, can correct 1-bit errors in up to 240 bits of data. Correcting bit errors is worthwhile, considering that in a study of IEEE 802.15.4 “around 50% of errors are isolated to a single bit, and 60% of burst errors are only two bits long” [2].
- We found 64 candidate polynomials for 16-bit CRC that can correct for 1- and 2-bit errors in less than 240 bits of data. Since TinyOS’ message format allows up to 232 bits of data, we recommend the use of one of these CRC polynomials.
- The number of generator polynomials that work for multiple-bit error correction is rather sparse as the message (data) size increases.

Future research can be pursued in the following directions: a thorough experimental evaluation of the SCC protocol, optimization of the tabular method such as that proposed in [4], modifying existing protocol stacks (e.g. TinyOS [7] or SunSPOT [6]) and investigating table space/time tradeoffs.

REFERENCES


Wireless Transmission Using Cooperation on Demand

Kamel Tourki  
Texas A&M University at Qatar  
Education City  
Doha, Qatar  
Email: kamel.tourki@qatar.tamu.edu

Mohamed-Slim Alouini  
Texas A&M University at Qatar  
Education City  
Doha, Qatar  
Email: alouini@qatar.tamu.edu

Mazen O. Hasna  
College of Engineering,  
Qatar University  
Doha, Qatar  
Email: hasna@qu.edu.qa

Abstract—Mobile users with single antennas can still take advantage of spatial diversity through repetition based cooperative transmission. In this paper, we consider a scheme in which the cooperation is triggered only if the source-destination channel is of an unacceptable quality. Therefore, the destination selects one relay out of a decoding set of relays for cooperation. We analyze the end-to-end outage probability in slow and fast fading environments and we evaluate the usefulness of relaying when the source acts blindly and ignores the decision of the selected relay whether it may cooperate or not. The performance in both environments are evaluated through analysis and simulations in terms of end-to-end outage probability and the number of active relays. The performance results show that the computer simulations based results coincide with our analytical results.

1. INTRODUCTION

In many wireless applications, users may not be able to support multiple antennas due to size, complexity, power, or other constraints. The wireless medium brings along its unique challenges such as fading and multiuser interference, which can be mitigated with cooperative diversity [1]–[7].

In [1], Laneman and Wornell proposed different cooperation protocols including fixed and adaptive relaying protocols. In the fixed relaying protocol, such as the amplify-and-forward and decode-and-forward protocols, the relays always help in forwarding the source information. Depending on the signal processing performed by the relay, relaying schemes can be classified as regenerative or non-regenerative (labeled in [1] as decode-and-forward and amplify-and-forward, respectively). With non-regenerative relaying, the relay node amplifies the received signal and then retransmits it. The main disadvantage of this kind of relaying is that the received noise and interference of the relay are amplified together with the signal. With regenerative relaying on the other hand, the relay detects and possibly decodes the source signal and then regenerates and retransmits it. This relaying scheme prevents noise propagation but requires more processing compared to non-regenerative relaying and may suffer from the probability of error detection of the signal at the relay. Cooperative relaying makes use of independent fading at source-destination and relay-destination channels.

The destination is then required to combine the signals from these different channels. In regenerative relaying, if the relay detection is correct, the destination receives the signal through two diversity paths. The authors in [1] also proposed adaptive relaying protocols including selection relaying and incremental relaying protocols. In selection relaying protocol, the relay forwards the information only if the amplitude of measured channel coefficient of source-relay link is larger than a certain threshold. With incremental relaying protocol, limited feedback from the destination is employed to indicate the success or failure of the direct transmission. Emamian and Kaveh proposed the cooperation as solution for combating shadowing [8] and Sendonaris et al. showed that cooperation among users can enlarge the capacity region of an uplink multiuser channel [9]. As mentioned in [10], the full spatial diversity benefits of the repetition based cooperative diversity algorithms come at the cost of decreasing bandwidth efficiency with the number of cooperative users. Instead, an alternative approach to improve bandwidth efficiency of the algorithms, based on space time codes is proposed in [1], [11], [12]. However, in regenerative relaying case, if the relay has a detection error, the effective SNR at the destination after combining is reduced significantly.

Therefore, some contributions on selective decode and forward cooperative communication under imperfect regeneration are presented in [11]–[15]. In [13], the source broadcasts its message to the relay and the destination subsequently and during the first transmission phase. In the second phase, if the instantaneous received SNR at the relay exceeds a certain threshold, the relay retransmits its received signal to the destination. This scheme is well known as time repetition coding in which the destination combines the received signal from the source and relay. Otherwise, if the source-relay channel quality is acceptable, the relay remains silent. This induces a rate loss with respect to non cooperative communication because the data is transmitted from different points in space, during different time slots. In [11], the relay is allowed to make errors and the authors opted for Distributed Space Time Coding (DSTC) with Orthogonal Frequency Division Multiplexing (OFDM) for a block fading channel. Therefore they proposed an optimal maximum likelihood (ML) decoder which exploits the knowledge of the error statistics at the relay and a suboptimal decoder when this knowledge is not available.

This work was supported by the Qatar National Research Fund (A member of Qatar Foundation).
A. Related Works

In [14], each relay decides whether to forward the source information or not by comparing the received signal power with a decision threshold. However, this threshold is selected in a way that the relay is able to forward only correctly decoded information. Other works study selective relaying schemes based on signal-to-noise ratio (SNR) to minimize the end-to-end bit error rate (BER) in cooperative regenerative relaying systems [12], [15], [16]. In [12], the authors consider a scheme in which a relay chooses to cooperate only if its source-relay channel is of an acceptable quality and we evaluate the usefulness of relaying when the source acts blindly and ignores the decision of the relays whether they may cooperate or not. While in [16], the cooperation decision is based on the source-relay, relay-destination and source-destination links. Analysis were conducted in [12], [16] considering the minimization of the end-to-end bit error rate as criteria. In [15], the authors considered a scheme in which the relay chooses to cooperate only if the source-destination channel is of an unacceptable quality. In that study, the authors consider a regenerative relay in which the decision to cooperate is based on a signal-to-noise ratio (SNR) threshold and consider the effect of the possible erroneously detected and transmitted data at the relay. A description of the proposed cooperative diversity scheme is presented, which is based on the output thresholded combining technique scheme introduced in [17]. Therefore depending on the SNR threshold level at the destination and at the relay, the cooperation can be triggered and the destination combines the received signals such as the relay can retransmit an erroneously decoded message. Expression for the end-to-end bit-error rate (BER) were derived and the authors looked at the optimal strategy to minimize this end-to-end BER at the destination for high SNR.

In literature, the researchers focused on different selective relaying schemes. In [18], the authors considered a selective orthogonal frequency division multiple access (OFDMA) relaying, where the relay selection is performed in a per-subcarrier manner. It means that several relays can be selected to assist the source transmission. Nevertheless, the authors presented a second scheme called selective OFDM relaying, where one best relay among L potential relays is selected to relay the entire OFDM packet. However with the selective OFDM relaying no diversity gain was obtained. Beres and Adve analyzed selection cooperation where only one relay was selected [19]. However, the selected relay is always cooperating and only large SNR regime is investigated. The novelty of this paper is detailed in the following.

B. Contributions

The main contributions of this paper are as follows. We develop cooperative diversity on demand (CDD) strategies for two proposed protocols based on repetition and distributed space-time coding using regenerative relays. Based on relay selection, our schemes reduce the number of forwarding relays to one. These protocols are used in slow and fast fading channels, respectively. The instantaneous source-destination SNR is used to enable cooperation. If the source-destination SNR is below a preselected threshold, the destination requests via a feedback path that the best relay decodes and forwards what it received from the source.

In a fast fading environment, the relay transmission is constrained by a threshold level as the destination. If the cooperation is triggered, the source retransmits its information. If a relay is selected, it re-encodes the received information from the source to form with this selected relay a distributed space-time coding system. We call this cooperation strategy blind cooperative diversity on demand (BC-CDD).

We determine the end-to-end outage probability expression for a given data rate $R$ in different environments with different architectures.

C. Organization of the Paper

The outline of the paper is as follows. In section II, we describe the system model and our proposed schemes. In section III, the end-to-end outage probability is derived in slow and fast fading environments. Finally, some selected simulation results are depicted in section IV while some concluding remarks are given in section V.

II. SYSTEM MODEL

In this section, we describe our proposed cooperative diversity scheme which is based on the output thresholded combining technique scheme and we note that only a selected relay from a cluster is targeted to cooperate. We investigate a system described in Figs 1-3. The source, destination and relays are denoted as $S$, $D$ and $r_k$ where $k \in \{1, ..., K\}$. We assume that each terminal is equipped with one antenna. The selected relay, $r_s$, cooperates if the direct transmission fails. We denote $h_{sr_k}$, $h_{sd}$ and $h_{rd}$ as the coefficients of the channels between the source ($S$) and the $k^{th}$ relay, the source and the destination ($D$), and the $k^{th}$ relay and the destination, modeled as flat fading and Rayleigh distributed with variances $\sigma^2_{sr_k}$, $\sigma^2_{sd}$ and $\sigma^2_{rd}$, respectively. Hence, we denote $\gamma_{sd} = E_s|h_{sd}|^2/N_0$ (resp. $\gamma_{sr_k} = E_s|h_{sr_k}|^2/N_0$) and $\gamma_{rd} = E_s|h_{rd}|^2/N_0$ the instantaneous received SNR at the destination (resp. at the $k^{th}$ relay) from the source and the $k^{th}$ relay, respectively, and $\bar{\gamma}_{sd} = \sigma^2_{sd}E_s/N_0$, $\bar{\gamma}_{sr_k} = \sigma^2_{sr_k}E_s/N_0$ are the average received SNR at the destination and the $k^{th}$ relay, respectively. We assume that the relays are close to each other and forming a cluster and we assume that the relays and the destination receive the same average SNRs $\bar{\gamma}_{sr_k}$ and $\bar{\gamma}_{rd}$ from the source and the relays respectively. Thus, we denote $\bar{\gamma} = \bar{\gamma}_{rd}$ and $\bar{\gamma}_{sr_k} = \bar{\gamma}_{sr_k}$ for all $k$.

Cooperation is triggered only when the direct transmission fails. The success/failure of the direct link is based on the instantaneous received SNR $\gamma_{sd}$. When it exceeds a threshold, we consider the direct transmission as a success. Otherwise, the cooperation is needed (second phase transmission) and

1. We assume short distances between the relays compared to the distances (S)-cluster, and cluster-(D), respectively.
the destination sends a binary feedback\footnote{As mentioned in [19], in a traditional MIMO system, with $m$ transmit antennas, selection requires $\log_2(m)$ bits of feedback. In our case, the destination feedback activates a specific bit $b = 1$ for $r_s$ while $b = 0$ for the remaining relays.} to the source and a selected relay $r_s$ requesting them to retransmit again where the destination chooses $r_s$ with the best instantaneous relay-destination channel by using the following rule

$$r_* = \arg \max_k (\gamma_{r,s})$$ \hspace{1cm} (1)

and we introduce $Y \triangleq \gamma_{r,s}$ for which we can express the cumulative density function (CDF) as

$$F_Y(y) = \frac{1}{2^{N}} \left(1 - e^{-y/\gamma} \right)^{K} \left(1 - e^{-y/\gamma} \right)^{K}.$$ \hspace{1cm} (2)

The probability density function (PDF) $f_Y(.)$ is then the derivative of the CDF $F_Y(.)$, i.e.

$$f_Y(y) = \frac{K}{\gamma} e^{-y/\gamma} \left(1 - e^{-y/\gamma} \right)^{K-1}.$$ \hspace{1cm} (3)

Using the binomial expansion we can rewrite $f_Y(.)$ as

$$f_Y(y) = \sum_{i=1}^{K} (-1)^{i-1} \left( \frac{K}{i} \right) \frac{y^{i}}{\gamma^{i}} e^{-iy/\gamma}.$$ \hspace{1cm} (4)

The cooperation scheme depends on channel environments. If the channel coefficients remain constants over two consecutive phases (slow fading conditions) and the cooperation is enabled, the source remains silent due to the source-destination link quality. However the selected relay has to decode and retransmit the received signal from the source using the same energy as shown in Fig. 2.

For fast fading channels, the fading amplitude changes between the two transmission phases and the source contributes in the second phase when the cooperation is needed by using distributed space time coding jointly with the relay $r_s$ as shown in Fig. 3.

Not that using selection cooperation, the transmission will be performed on orthogonal channels and the required bandwidth will be only doubled while $K$ relays can potentially contribute to the transmission.

III. OUTAGE ANALYSIS

In our scheme, cooperation is triggered depending on the capacity of the link between the source and the destination. We introduce the decoding set, $D(s)$, as the set of relays that decoded the information correctly during the first phase as depicted in figure 1. Thereby, $r_k \in D(s)$ if the corresponding $s-r_k$ link has a capacity above the required rate $R$. Therefore,

$$C_{r_k} = \frac{1}{2} \log \left(1 + \left|h_{s,r_k}\right|^2 \frac{E_k}{N_0} \right) \geq R,$$ \hspace{1cm} (5)

where the factor $1/2$ models the two time slots required for relaying given the half-duplex constraint, $N_0$ is the noise power spectral density and as mentioned above, the selected relay transmits using the same energy as the source, $E_s$. Therefore $P_r[r_k \in D(s)]$ is expressed as

$$P_r[r_k \in D(s)] = e^{-k \Phi/\gamma_{sr}}.$$ \hspace{1cm} (6)

where $\Phi = 2^{2R} - 1$. Hence, the probability of a decoding set with cardinality equal to $k$ is derived as

$$P_r[[D(s)] = k] = \left( \frac{K}{k} \right) e^{-k \Phi/\gamma_{sr}} \left(1 - e^{-\Phi/\gamma_{sr}} \right)^{K-k}.$$ \hspace{1cm} (7)

A. Slow Fading Channels

As mentioned before, our proposed cooperative scheme depends on the channel environment. For slow fading channels, we assume that the channel coefficients remain constant during two consecutive transmission periods. The destination eventually combines the received signals from the source in the first phase and from the relay in the second phase. The outage event for the cooperative on demand scheme using the selection relaying in the slow fading environment is

$$\left( \{\gamma_{sd} < \Phi\} \cap \{\gamma_{sr} < \Phi\} \right) \cup \left( \{\gamma_{sd} < \gamma_{sr} < \Phi\} \cap \{\gamma_{sd} + \gamma_m < \Phi\} \right) \cap \{\gamma_{sd} > \Phi\}.$$ \hspace{1cm} (8)

where $\Phi' = 2^{R} - 1$. Therefore, the end-to-end outage probability $P_{\text{out}}$ can be expressed as

$$P_{\text{out}} = \left( \prod_{k=1}^{K} P_{sr_k} \right) P_{\text{out}}^I + \sum_{k=1}^{K} P_{r}[[D(s)] = k] P_{\text{out}}^I,$$ \hspace{1cm} (9)

where

$$P_{r}[[D(s)] = k]$$ \hspace{1cm} (10)

is defined in (7), and

$$P_{\text{out}}^I = P_r[\gamma_{sd} < \Phi', \gamma_{sd} + \gamma_m < \Phi] = \int_{0}^{\Phi'} \int_{0}^{\Phi-x} f_{Y}(y)dydx,$$ \hspace{1cm} (11)

where a new exponentially distributed variable $X \triangleq \gamma_{sd}$ is introduced.

1) Equal Channel Gains: For equal sub-channel gains where $\gamma_{sd} = \gamma$, $P_{\text{out}}^I$ defined in (11) is given by

$$P_{\text{out}}^I = K \left[ 1 - e^{-\Phi'/\gamma} - \frac{\Phi'}{\gamma} e^{-\Phi'/\gamma} \right] + \sum_{i=2}^{K} \left(-1\right)^{i-1} \left( \frac{K}{i} \right) \times \left[ 1 - e^{-\Phi'/\gamma} - \frac{\Phi}{\gamma} e^{-\Phi'/\gamma} \right] \left(1 - e^{-(i-1)\Phi'/\gamma} \right).$$ \hspace{1cm} (12)

2) Unequal Channel Gains: For unequal sub-channel gains where $\gamma_i \neq \gamma_{sd}$ $\forall i = 1 \cdot K$, $P_{\text{out}}^I$ defined in (11) is given by

$$P_{\text{out}}^I = \sum_{i=1}^{K} \left(-1\right)^{i-1} \left( \frac{K}{i} \right) \left[ 1 - e^{-\Phi'/\gamma_{sd}} - \frac{\gamma_i e^{-\Phi'/\gamma}}{\gamma - i \gamma_{sd}} e^{-\Phi'/\gamma_{sd}} \right] \left(1 - e^{-(i-1)\Phi'/\gamma} \right).$$ \hspace{1cm} (13)
B. Fast Fading Channels

During the first phase, the destination receives the transmitted data from the source. When the cooperation is triggered, the source is allowed to retransmit because the source-destination channel coefficient, $h_{sd}$, will change in the second phase ($h_{rd}^{n+1} \neq h_{sd}^n$). The outage event for the cooperative on demand scheme using the selection relaying in the fast fading environment is

$$
\left( \{ \gamma_{sd}^f < \Phi' \} \cap \{ \gamma_{sd}^f + \gamma_{id}^f < \Phi \} \right) \cup \left( \{ \gamma_{sd}^f < \Phi' \} \cap \{ \gamma_{sd}^f + \gamma_{id}^f + \gamma_m < \Phi \} \cap \{ \gamma_{sr} < \Phi \} \right),
$$

where $\gamma_{sd}^f$ and $\gamma_{id}^f$ are the instantaneous received SNRs by the destination from the source during the first and the second phase, respectively. Therefore, the end-to-end outage by the destination from the source during the first and the second phase, respectively. Hence, the end-to-end outage probability $P_{out}$ keeps the same expression as in (9). Then, we introduce a new random variable $Z \left( \frac{\gamma_{id}}{\gamma_{sd}} \right) = X \left( \frac{\gamma_{id}}{\gamma_{sd}} \right) + Y \left( \frac{\gamma_{id}}{\gamma_{md}} \right)$. The corresponding PDF $f_Z(.)$ is expressed as

$$
f_Z(z) = \sum_{i=1}^{K} (-1)^{i-1} \left( \frac{K}{i} \right) \frac{i}{\gamma_{sd} - \gamma} \left( e^{-z/\gamma_{sd}} - e^{-iz/\gamma} \right)
$$

Based on the outage event expression in (14), $P_{out}$ and $P_{out}^I$ are given by

$$
P_{out} = P_r \left[ \gamma_{sd} < \Phi', \gamma_{sd}^f + \gamma_{id}^f < \Phi \right] = 1 - e^{-\Phi'/\gamma_{sd}} - \frac{\Phi'}{\gamma_{sd}} e^{-\Phi/\gamma_{sd}},
$$

and

$$
P_{out}^I = P_r \left[ \gamma_{sd}^f < \Phi', \gamma_{sd}^f + \gamma_{id}^f < \Phi \right]
= \int_0^{\Phi'} f_X(x) \int_0^{\Phi - x} f_Y(y) dy dx,
$$

respectively.

1) Equal Channel Gains: For equal sub-channel gains where $\gamma_{id} = \gamma$, $f_Z(.)$ defined in (15) can be rewritten as

$$
f_Z(z) = \frac{K}{\gamma} z e^{-z/\gamma} + \sum_{i=2}^{K} (-1)^{i-1} \left( \frac{K}{i} \right) \frac{i}{\gamma(i-1)} \times \left( e^{-z/\gamma} - e^{-iz/\gamma} \right)
$$

So, $P_{out}$ defined in (17) can be computed as

$$
P_{out} = K \left( 1 - e^{-\Phi'/\gamma} \right) - \frac{\Phi'}{\gamma} \left( 1 + \frac{\Phi'}{\gamma} \right) e^{-\Phi'/\gamma} + \frac{K \Phi'^2}{2\gamma^2} \times e^{-\Phi'/\gamma} + \sum_{i=2}^{K} (-1)^{i-1} \left( \frac{K}{i} \right) \left( 1 - e^{-\Phi'/\gamma} - \frac{i}{i-1} \frac{\Phi'}{\gamma} e^{-\Phi'/\gamma} \right)
- \sum_{i=2}^{K} (-1)^{i-1} \left( \frac{K}{i} \right) \frac{e^{-\Phi'/\gamma}}{(i-1)^2} \left( 1 - e^{-(1-i)\Phi'/\gamma} \right)
$$

2) Unequal Channel Gains: For unequal sub-channel gains where $\gamma_{id} \neq \gamma_{id}^f \forall i = 1: K$, and using the PDF expression in (15), $P_{out}^I$ defined in (17) can be written as

$$
P_{out}^I = \sum_{i=1}^{K} (-1)^{i-1} \left( \frac{K}{i} \right) \left( 1 - e^{-\Phi'/\gamma_{sd}} - \frac{\Phi'}{\gamma_{rd} - \gamma} e^{-\Phi/\gamma_{sd}} \right)
- \sum_{i=1}^{K} (-1)^{i-1} \left( \frac{K}{i} \right) \left( 1 - e^{-\Phi'/\gamma} - \frac{i\Phi'}{\gamma_{rd} - \gamma} e^{-\Phi/\gamma} \right)
$$

IV. PERFORMANCE RESULTS

A. Network Geometry

We anticipate that cooperation will perform differently as function of the positions of the users with respect to destination. Hence we study two different network geometries, denoted by symmetric network (SN) and asymmetric or linear network (LN). In the LN case, we model the path-loss, i.e., the mean channel powers $\sigma_{ij}$, as a function of the relays cluster position $d_k$ by

$$
\sigma_{sd} = 1, \sigma_{sr} = d^{-\nu}, \sigma_{rd} = (1 - d)^{-\nu}
$$

where $\nu$ is the path loss exponent and $0 < d (= distance_{cluster}) < 1$. The distances are normalized by the distance $d_{sd}$. In these coordinates, the source can be located at $(0,0)$, the destination can be located at $(1,0)$, without loss of generality, and the relays are located at $(d,0)$ [20]. In the SN case, $h_{sd}$ and $h_{rd}$ are drawn with the same unit-variance (equal sub channel gains), but considering that the source and the relays cluster are close to each other, we set $\sigma_{sr}^2 = 16$ for all relays.

B. Simulation Results

In this section, we evaluate the performance of our scheme in terms of the end-to-end outage probability at the destination as function of the SNR $E_b/N_0$ for a targeted transmission rate $R = 1$ (bits/s/Hz), and the number of active relays in phase II. All figures show that a full diversity order is achieved. Figure 4 (resp. figure 8) depicts the outage analysis in SN networks where all nodes evolve in a slow (resp. fast) fading channel. The outage performance of the proposed relay selection is improved as the parameter values of $K$ increase and the achieved diversity order is $K + 1$, and $K + 2$ in Fig. 8. Figs 5-7 depict the outage analysis for slow fading environment with LN architecture where $K = 1$, $K = 2$ and $K = 4$, respectively. In particular, the system performs better when the relay cluster is located at $d_2 = 0.5$. Similar results are shown in figures 9-11 where the source retransmission in the second phase enhances the diversity order.

The relay cluster behavior is reduced to its selected relay cooperation where a full diversity order is achieved, temporal overhead reduced and power saving is guaranteed.
V. CONCLUSIONS

In this work, we studied the outage analysis of a cooperation on demand based on the output thresholded MRC scheme in different environment. Increasing the number of selected relays leads to a linear influence on the spectral efficiency, so to larger temporal overhead for the transmission. However, selecting only one relay out of $K$ active relays doubles the spectral efficiency and keeps a full diversity order, $(K+1)$ and $(K+2)$ in slow fading and fast fading environments, respectively.

REFERENCES


Fig. 1. Broadcasting phase (Phase I) in a multi-relay network with selection relaying where $K = 7$ and ($|D(s)| = 4$).

Fig. 2. Selected relay transmission (Phase II) in a slow fading environment.

Fig. 3. Cooperative transmission (Phase II) in a fast fading environment.
multiple relays are candidates in a slow fading environment.

Fig. 4. End-to-end outage probability versus SNR in the SN network where multiple relays are candidates in a fast fading environment.

Fig. 5. End-to-end outage probability versus SNR in the LN network, which is evolving in a slow fading environment, where four relays are active in the cluster which is located at \( d_1 = 0.1 \), \( d_2 = 0.5 \) and \( d_3 = 0.9 \), respectively.

Fig. 7. End-to-end outage probability versus SNR in the LN network, which is evolving in a slow fading environment, where only one relay is located at \( d_1 = 0.1 \), \( d_2 = 0.5 \) and \( d_3 = 0.9 \), respectively.

Fig. 8. End-to-end outage probability versus SNR in the SN network where multiple relays are candidates in a fast fading environment.

Fig. 9. End-to-end outage probability versus SNR in the LN network, which is evolving in a fast fading environment, where only one relay is located at \( d_1 = 0.1 \), \( d_2 = 0.5 \) and \( d_3 = 0.9 \), respectively.

Fig. 4. End-to-end outage probability versus SNR in the SN network where multiple relays are candidates in a slow fading environment.

Fig. 10. End-to-end outage probability versus SNR in the LN network, which is evolving in a fast fading environment, where two relays are active in the cluster which is located at \( d_1 = 0.1 \), \( d_2 = 0.5 \) and \( d_3 = 0.9 \), respectively.
Fig. 11. End-to-end outage probability versus SNR in the LN network, which is evolving in a fast fading environment, where four relays are active in the cluster which is located at $d_1 = 0.1$, $d_2 = 0.5$ and $d_3 = 0.9$, respectively.

Fig. 6. End-to-end outage probability versus SNR in the LN network, which is evolving in a slow fading environment, where two relays are active in the cluster which is located at $d_1 = 0.1$, $d_2 = 0.5$ and $d_3 = 0.9$, respectively.
Power Consumption in Wireless Sensor Network Using Network Coding

Nagajothy.M, Dr.S.Radha, Member, IEEE

Abstract—Power management is the key issue in the design and operation of wireless network applications like sensor networks, pervasive computing and ubiquitous computing where the network is primarily driven by battery-powered embedded devices. This paper studies network coding as power minimization technique. Network coding is a new research area that may have interesting applications in practical networking systems. With network coding, intermediate nodes may send out packets that are linear combinations of previously received information. Thus network coding reduces the power consumption by minimizing the number of transmissions required to communicate a given amount of information across the network. Hence it improves the lifetime of the network when network coding is being applied. This paper explains what is network coding and the benefits of network coding. Evaluation results indicate that the proposed solution is more efficient than no-network coding solution while still meeting the required lifetime constraints.

Index Terms—Network coding, Power control, Reliability, Wireless sensor networks.

I. INTRODUCTION

Communication networks today share the same fundamental principle of operation. Whether it is packets over the Internet, or signals in a phone network, information is transported in the same way as cars share a highway or fluids share pipes. That is, independent data streams may share network resources, but the Information itself is separate. Routing, data storage, error control, and generally all network functions are based on this assumption. Network coding is a recent field in information theory that breaks with this assumption. Instead of simply forwarding data, nodes may recombine several input packets into one or several output packets.

In today’s practical communication networks such as the Internet, information delivery is performed by routing. A promising generalization of routing is network coding. The potential advantages of network coding over routing include resource (e.g., bandwidth and power) efficiency, computational efficiency, and robustness to network dynamics. Network coding is also very well suited for environments where only partial or uncertain information is available for decision making. Similar to erasure coding, successful reception of information does not depend on receiving specific packet content but rather on receiving a sufficient number of independent packets. Linear combining requires enhanced computational capabilities at the nodes of the network. However, according to Moore’s law, processing is becoming less and less expensive. The bottleneck has shifted to network bandwidth to support the ever growing demand in applications and QoS guarantees over large unreliable networks. Thus network coding utilizes cheap computational power to increase network efficacy.

Recent technological advancements in wireless communications are fundamentally changing the manner by which devices communicate with one another. Modern wireless devices build networks on their own and aid each other in passing information to any device in the network. Many are battery powered, thus energy conservation is critical to maximizing their useful lifetime. We will show that network coding, a technique originally introduced to maximize throughput, can be used as a technique to save energy by minimizing the number of transmissions needed to deliver a set of packets.

II. NETWORK CODING AND ITS BENEFITS

The basic idea of network coding is illustrated in Fig. 1 where nodes A, B and C share the common wireless medium as in [1]. Consider a scenario where nodes A and C have information to exchange. Due to the channel constraints only one node can transmit at any given time. Node A sends its packet (p1) to relay node B. Node B forwards this packet to node C. Similarly, node C sends its packet (p2) to node B which in turn forwards it to node A as shown in the fig.1. This involves a total of four transmissions. Now consider the scenario where network coding is applied to reduce the number of transmissions.
Nodes A and C transmit packets to central node B sequentially (two transmissions). Node B, instead of directly forwarding each packet to its destination, XOR’s the two packets and broadcasts the result as a single packet in the shared medium as shown in the figure. Both nodes A and C know their own packet (p1 and p2, respectively) that originated from them. They can therefore retrieve the unknown packet by XORing the known packet with broadcast packet. This entire process takes exactly three transmissions as opposed to four transmissions as discussed above. In a simplistic model this technique will result in 25% less energy consumption.

The most famous example of this benefit was given by Ahlswede et al. [1], who considered the problem of multicast in a wire line network. Their example, which is commonly referred to as the butterfly network (Fig. 2), features a multicast from a single source to two sinks, or destinations.

Assume that we multicast two data bits b1 and b2 from the source node S to both the nodes Y and Z in the acyclic network depicted by Fig.2(b). Every channel carries either the bit b1 or the bit b2 as indicated. Fig.2(a) depicts a different way to multicast two bits from the source node S to Y and Z. In this way, all the 9 channels in the network are used exactly once. If the same communication objective is to be achieved simply by bit replication at the intermediate nodes without coding, at least one channel in the network must be used twice so that the total number of channel usage would be at least 10. Thus, coding offers the potential advantage of minimizing both latency and energy consumption, and at the same time maximizing the bit rate.

**III. NETWORK CODING APPLIED FOR CIRCULAR AND GRID NETWORKS**

**A. Circular Networks**

Consider n nodes placed at equal distances around a circle as depicted in Fig.3. Assume that each node can successfully broadcast information to its two neighbours. It has been analysed earlier as in [7] that without network coding it needs n-2 transmissions and with network coding it needs only n-1/2 transmissions which is shown in Fig.4.

**B. Grid Networks**

Consider a wireless ad-hoc network with n nodes. Let n = m^2. Assume that every node is a source that wants to broadcast information to all n nodes. The nodes are placed on the vertices of a rectangular grid and each node can successfully broadcast information to its four nearest neighbors as shown in Fig.5.
It has been analysed earlier as in [7] that without network coding it needs $n^2/3$ transmissions and with network coding it needs only $n^2/4$ transmissions which is shown in the fig.6.

IV. POWER MINIMIZATION USING NETWORK CODING

For simulating the wireless sensor environment, an area of size 200*200 m$^2$ was assumed. The average power consumed by the radio is modelled as in [8],

$$P_{radio} = N_{tx} [P_{tx}(T_{on} + T_{off})] + P_{out}(T_{on} + T_{off}) + N_{rx}[P_{rx}(T_{on} + T_{off})]$$

(1)

where $N_{tx/rx}$ is the average number of times per second that the transmitter/receiver is active; $P_{tx/rx}$ is the power consumed by the transmitter/receiver; $P_{out}$ is the output transmit power; $T_{on}$ the transceiver start up time; and is the actual data transmitting/receiving time equal to $L/R$, where $L$ is the packet length in bits and $R$ is the data rate in bits per second. The following assumptions were made as shown in table 2 to obtain the results.

<table>
<thead>
<tr>
<th>TABLE 1</th>
<th>SIMULATION SETUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radio</td>
<td>CC1000</td>
</tr>
<tr>
<td>Data Rate</td>
<td>76.8 Kbps</td>
</tr>
<tr>
<td>Modulation Type</td>
<td>FSK</td>
</tr>
<tr>
<td>Receive Power</td>
<td>29 mW</td>
</tr>
<tr>
<td>Transmit Power at 0 dbm</td>
<td>42 mW</td>
</tr>
<tr>
<td>Sleep Power</td>
<td>100 $\mu$W</td>
</tr>
<tr>
<td>Initial Power of each node</td>
<td>200 mW</td>
</tr>
</tbody>
</table>

For circular and grid networks the total amount of power consumed by the total number of nodes in the network has been analysed and plotted as shown in fig.7 & fig.8 respectively. It has been observed that the amount of power utilized by the network when network coding is applied is less when compared to conventional routing (without coding) for both circular and grid networks.
V. IMPROVEMENT IN LIFETIME

An important performance indicator for the wireless sensor network is the node/network lifetime. The network life time has been referred to as the fraction of surviving nodes in the network. Every node in the network can be taken to have initial battery power $P_{battery}$. For a given data rate $R_b$ the time taken to transmit one packet is $L/R_b$. So to transmit one packet the total amount of power required would be written as [9],

$$P_{packet} = P_b L / R_b$$  \hspace{1cm} (2)$$

Since nodes transmit packets with average rate $\lambda$, average energy depleted per second is $\lambda \cdot P_{packet}$. So finally the total time taken for the battery power to be exhausted can be written as

$$\tau = \frac{P_b R_b}{\lambda L P_t}$$  \hspace{1cm} (3)$$

where $\tau$ is the lifetime of the node in seconds, $R_b$ is the data rate, $L$ is the packet length which is taken as 1000 bits, $\lambda$ is the packet arrival rate given as 0.5 packet/s and $P_t$ is the transmitted power.

It has been analysed that as we increase the transmit power of the node, more energy would be spent by that node and hence the lifetime of each node decreases. However if the node participates in network coding then the remaining energy of that node after coding will be more than the node which does not participates in network coding. This justifies that whenever network coding is applied then it can reduce the power consumption of the node since the number of transmissions required is reduced and hence it improves the lifetime of the same node which is clearly shown in the fig.9.

VI. CONCLUSION

In this paper, the use of network coding for wireless mesh networks is studied. Through extensive simulations, it is investigated that how this transmission paradigm behaves under a variety of scenarios. Performance evaluation indicates that only minimum amount of power is required to transmit per unit of information in a circular and grid network by applying network coding since it needs less number of transmissions thereby increasing the lifetime of each individual node. This work indicates that there is a potential for significant benefits, when deploying network coding over a practical wireless network environment.

REFERENCES


First A. M.Nagajothy graduated from Shri Angalamman College of Engineering and Technology, Anna university, Chennai in Electronics and Instrumentation Engineering in the year of 2005. She is doing her Master’s degree in Communication Systems in SSN college of Engineering, Anna University, Chennai during the year 2007-2009. She has 2 publications in National Conference in the area of Wireless Sensor Networks.

Second B. S.Radha graduated from Madurai Kamraj University, in Electronics and Communication Engineering during the year 1989. She obtained her Master degree in Applied Electronics from Government of Technology, Coimbatore and PhD degree in the area of Mobile Ad Hoc Networks from College of Engineering, Gunidy, Anna University, Chennai. At present she is working as Professor in the Department of ECE, Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam, India. She has 33 publications in International and National Journals and conferences in the area of Mobile Ad hoc Network. Presently she is guiding 9 research scholars in the area of security in mobile ad hoc networks, sensor networks. Her current areas of research are security and architecture issues of mobile ad hoc networks and sensor networks. She received IETE – S K Mitra Memorial Award in October 2006 for the Best Research Paper published in the IETE Journal of Research and CTS – SSN Best Faculty Award – 2007 for the outstanding performance for the academic year 2006-07.
DCOSS Workshop:

International Workshop on Robotic Wireless Sensor Networks
Actuated Sensing Systems for High Fidelity Environmental Monitoring

Diane Budzik, Andrew D. Parker, Maxim A. Batalin and William J. Kaiser
Center for Embedded Networked Sensing, University of California, Los Angeles

Abstract—Many environmental sensing applications such as solar light radiation and carbon dioxide flux require observing dynamic spatio-temporal phenomena spread over large spatial domains. Static Sensing, Deterministic Actuated Sensing, Adaptive Sensing, and Multiscale Sensing have all been used to monitor such phenomena. In order to compare the performance of these algorithms for dynamic spatio-temporal phenomena, we have implemented versions of each of these algorithms. We use the application of monitoring light intensity in a forest understory as the basis for comparison. Since robotic platforms have a variety of acceleration and velocity profiles, we compared the performance of these algorithms in terms of run time and the number of observations made. Since Deterministic Actuated Sensing, Adaptive Sensing, and Multiscale Sensing all use mobile robots to make observations, we performed extensive empirical evaluations in simulation using field data to analyze how robot speed impacts the performance of each algorithm. We also studied how the performance of Multiscale Sensing changes when it is implemented using multiple robots.

I. INTRODUCTION

Environmental monitoring applications often involve observing phenomena characterized by spatial and temporal dynamics covering large spatial domains. A few examples of such phenomena include solar light radiation, carbon dioxide flux, humidity, nitrate distribution in soil, and dissolved oxygen [1]–[3]. Measuring solar light radiation provides information about plant distribution in a forest, e.g., distribution of native vs. invasive non-native plants, which is an indicator of the health of an ecosystem. The bracken fern, which is carcinogenic to some animals, is an example of an invasive plant that can force out native species [4].

Static sensors have been used extensively for a broad range of environmental monitoring applications such as obtaining accurate hydrological models and avalanche prediction [5], studying the microclimate of a redwood tree [6], and the behavior of seabird colonies [7]. Static sensor networks perform accurate reconstruction of temporally varying phenomena; however, their inability to change position leads to a spatial phenomenon reconstruction. Characterizing spatially varying phenomena with only static sensors requires an impractically large number of sensors to be distributed across the complete spatial extent of the environment. Therefore, actuated sensing - mobile robots carrying the required sensors - is needed to accurately measure complex spatio-temporal dynamics over large spatial domains.

Deterministic Actuated Sensing, i.e., a raster scan, is a conventional default choice for sensing environmental phenomena when no previous models exist. In a raster scan, a mobile robot actuates along a predetermined, uniform path at predetermined locations. The density of the raster scan can be adjusted according to the desired fidelity. Many application scientists use Deterministic Actuated Sensing to collect data [8], [9]. If the phenomena is dynamic in both space and time over an expansive area, however, the sensing latency resulting from actuation may become too large to achieve the desired sensing fidelity.

Several algorithms, generally termed Adaptive Sensing, have been proposed to address the problem of sensing latency by reducing the number of observation locations while still achieving high fidelity [10]–[12]. For example, Rahimi et al. [10] proposed a multi-step approach that varies observation densities at each step. In the first step, a mobile robot performs a coarse scan of the complete environment to extract regions of high phenomenon variability. With each subsequent step, overall sensing fidelity is improved by making higher density observations in regions with high variability. Such adaptive sensing algorithms perform well in cases where the environmental phenomena has high spatial variation but low temporal variation. The sensing latency of adaptive sensing is still too large, however, for phenomena that is dynamic in both space and time.

Another approach that significantly reduces sensing latency uses a combination of static sensors and mobile robots [13], [14]. A set of \( n \) static sensors is deployed discretizing the environment into \( n \) regions with one sensor per region. These sensors monitor their respective regions for events of interest such as a high concentration of phenomena or dynamic phenomena. In this approach, the static sensors may only act as triggers and need not be used for making observations. When an event is detected, the mobile robot is notified and tasked to make observations only in the region of the sensor triggered by the event. This approach reduces the sensing latency such that it can be used for sensing dynamic spatio-temporal phenomena. Events occurring outside the range of a static sensor, however, may be missed in this approach. Thus, the performance of the system depends on the number of sensors, i.e., the level of discretization of the environment. This results in a trade-off between the number of static sensors and the reconstruction fidelity of the phenomena.

This motivated Singh et al. [15] to propose a two-tier Multiscale Sensing (MS) approach to reduce the large sensing latency that results from actuation. We have implemented a version of the MS approach for this study. In their approach, the first tier uses a static low fidelity, high spatial coverage sensor providing real-time “global” information about the environment. The second tier, using actuated sensing, exploits the information from the first tier to perform informed actuated sensing in the regions of interest. In this paper, an in depth comparison of the performance of Deterministic Actuated Sensing, Adaptive Sensing, and Multiscale Sensing for dynamic spatio-temporal phenomena is analyzed. We use the important application of monitoring light intensity in a forest understory as the basis for comparison. Since robotic platforms have a variety of acceleration and velocity profiles, we com-
Algorithm: Adaptive Sensing

Input: $d$, $dRoot$, minVar, minDim, root
Output: Loc, Val

begin
1. $[loc, val] \leftarrow observeStratum(root, dRoot);
   Loc $\leftarrow arr(loc);
   Var $\leftarrow arr(val);
2. rootVar $\leftarrow getVarStratum(root, Loc, Val);
   Q $\leftarrow arr([root, rootVar]);$
   while notEmpty(Q) do
3.   $[S, svar] \leftarrow head(Q);
   Q $\leftarrow delete(head(Q));$
4.   foreach $s \in S$ do
5.     if $S == root$ then
6.       $[loc, val] \leftarrow reuseObs(s, Loc, Var);
7.     else
8.       $[loc, val] \leftarrow observeStratum(s, d);
   Loc $\leftarrow append(Loc, loc);
   Val $\leftarrow append(Val, val);
   totalVar $\leftarrow getVarStratum(s, Loc, Val);
9.     if totalVar $\geq$ minVar AND
10.    dim(s) $\geq$ minDim then
11.       $Q \leftarrow revSort(append(Q, [s, totalVar]));$
12. return (Loc, Val);

Algorithm 1: Adaptive Sensing

To evaluate Adaptive Sensing (AS), we implemented a modified version of the algorithm described in [10]. Our AS algorithm is outlined in Algorithm 1. This implementation produces a recursive dyadic partitioning (RDP) of the region under investigation, similar to that in [11], so that the observation locations are concentrated in regions that have high variability.

AS begins with a coarse, uniform scan of the complete region under investigation (c.f. Line 1 of Algorithm 1). The observation density of the coarse scan is the user-specified input parameter, dRoot. Fig. (1d) illustrates an example of a coarse scan with an observation density of 245 pixels, or about 191 cm. Note that in this implementation of AS, observations are made on the made of the environment of interest as seen in Fig. (1d). Observing the edges of the environment improves phenomena reconstruction, thus, Deterministic Actuated Sensing also follows the same procedure. After the initial coarse scan, the variance of the complete environment or root stratum, root, is calculated (c.f. Line 2 of Algorithm 1). Next, the root stratum is partitioned into four substrata (c.f. Line 4 of Algorithm 1). The observations made during the coarse scan are used to calculate the variance for each of the four substrata (c.f. Line 5 of Algorithm 1). For each stratum, if the calculated variance is greater than the minimum variance, minVar, and if the dimension of the stratum are the minimum dimension, minDim, then that stratum and its corresponding variance are added to the variance queue, Q, and the queue is reverse sorted (c.f. Line 7 and Line 8 of Algorithm 1). The minimum variance is defined as 5% of the maximum variance of a given phenomena field. In Fig. (1), the minimum variance was 1500. The dimension of a stratum is defined as the width and height of the stratum. The minimum dimension in this example is 31 pixels x 31 pixels since the observations density, $d = 31$ pixels.

After each of the four child strata has been evaluated, the main loop repeats: the stratum with the largest variance is popped from the queue, partitioned, scanned with observation density, $d$, and evaluated (c.f. Line 3 through Line 8, of Algorithm 1). The paths shown in Fig. (1e) have an observation density of 31 pixels, or about 24 cm. When scanning the child stratum, the pattern of the locations of the new observations are offset based on the depth of the child stratum in order to avoid redundantly sampling the same locations from a parent stratum. The overall resulting pattern of locations approximates a maximin distance design, which is a design that given a number of points and an observation window, maximizes the minimum distance between all point pairs [16]. An example of this is shown in Fig. (1e) with a more detailed view shown in Fig. (1f). The main loop of the algorithm repeats until the variance queue of candidate strata, Q, has been exhausted. As output, the AS algorithm returns a list of observation locations and corresponding values, Loc and Val, respectively.

It is important to note that the observation density of the root stratum and the observation density of subsequent strata often differ. This flexibility allows the user to decide how much time to allocate to the root stratum versus the child strata. For instance, when the phenomena is known to have localized features, e.g., smooth regions separated by boundaries as is the case in Fig. (1a), it is advantageous to perform a quick, coarse scan to find the smooth regions. As a result, regions...
Algorithm: Multiscale Sensing

Input: \( r, h, v, b, o, i \)

Output: \( \text{Loc}, \text{Val} \)

begin
1. \( T \leftarrow \text{tier1TaskExtraction} \);
2. \( PT \leftarrow \text{taskPrioritization}(T) \);
   for \( 1 \leq j \leq \text{num}(T) \) do
   3. \( TC_j \leftarrow \text{taskAllocation}(PT(j), r) \);
   4. \( \text{loc}_j, \text{val}_j \leftarrow \text{taskObservation}(TC(j), h, v, b, o, i) \);
return \( \text{Loc}, \text{Val} \);
end

Algorithm 2: Multiscale Sensing

of low variance do not require further observation, allowing more observations to be made in regions of high variance.

D. Multiscale Sensing

Multiscale Sensing (MS) was first introduced in [15]. A two-tier instance of Multiscale Sensing is implemented here. Algorithm 2 provides an outline for the Multiscale Sensing algorithm. The first tier sensor (an imager) provides regions of interest referred to as tasks, for mobile robots to service (c.f. Line 1 of Algorithm 2). For the light intensity application the sunlit bright regions are the regions of interest (c.f. Fig. (1b)). Note that while a single sensor comprises the first tier in this case, a set of static sensors whose data is then interpolated could also make up the first tier. Task extraction is performed next. Gray scale images are first converted to binary (black and white) images via a threshold found using Otsu’s method [17]. This method chooses a threshold that minimizes the intra-class variance of the pixel values above and below the threshold. Then, edge detection is performed on these binary images by tracing the exterior edges of the white objects. Fig. (1g) illustrates this task extraction procedure on the image specified in Fig. (1a).

We perform task prioritization next. During task prioritization, a heuristic is used to determine which region of interest (task) to observe. We use a greedy heuristic that prioritizes tasks based on the total area to observe (c.f. Line 2 of Algorithm 2). Tasks with an area smaller than an experimentally determined threshold are discarded as noise, i.e., not having large enough areas to constitute significant regions of interest. Next, task allocation is performed by assigning tasks to mobile robots for servicing, i.e., making observations. MS takes as input a parameter \( r \), which specifies the number of robots available for servicing tasks. For a set of prioritized tasks, a set of assignments, \( a \), is computed:

\[
\forall a \in A^a = \arg \max_{n=(1,\ldots,|R|)} U(r_n, j),
\]

where \( j \) is the next unassigned task according to our greedy area heuristic and \( U(r_n, j) \) is the \( n^{th} \) robot utility value, i.e., in our implementation it is the distance of the \( n^{th} \) robot to the task, for accomplishing \( j \) (servicing the task). Essentially, the algorithm assigns the highest priority task to the nearest robot. Note that each robot services each task until completion; Therefore, a robot is only available for servicing a new task after it has finished servicing its prior task. Fig. (1h) and Fig. (1i) illustrate this task allocation step for the phenomena shown in Fig. (1a). Fig. (1h) shows the first task that was selected for observation, which in this implementation is the largest task of all the tasks extracted in Fig. (1g), (c.f. Line 3 of Algorithm 2). Fig. (1h) also shows the path MS takes as it makes observations. MS begins sensing in the upper left corner of each task and makes observations via a uniform scan within the task being serviced (c.f. Line 4 of Algorithm 2). MS takes as input several parameters \( h, v, b, o, i \) and \( i \) that specify the densities and locations in which MS makes observations. The parameters \( h \) and \( v \) specify the observation density in the vertical and horizontal direction, respectively. Parameter \( b \) specifies the observation density in the vertical direction along the task boundary. Lastly, \( o \) and \( i \) specify how far inside and outside of the task boundary to make observations. In this implementation of Multiscale Sensing, \( h, v, b, o, i \) and \( i \) are in the units, pixels. In this example \( h = 25 \) pixels, \( v \) and \( b = 20 \) pixels, and \( o \) and \( i = 10 \). Fig. (1i) shows the second task that was selected for observation and the locations where MS makes observations. The locations where MS makes observations and the values of those observations are the output of the MS algorithm, \( \text{Loc} \) and \( \text{Val} \), respectively.

III. EXPERIMENTS AND ANALYSIS

In this section we compare the performance of several environmental monitoring algorithms for the important application of light intensity monitoring through extensive simulations using field data. We collected a series of images throughout the day in a mixed conifer forest at the James San Jacinto Mountain Reserve in Southern California [18]. These images were collected using a down-looking camera capturing an area approximately 6 m in length by 4 m in width. We used two images, one collected at approximately 8 a.m. and one collected at approximately 10 a.m., that are representative of the spatial variations occurring in the observed environment.

For each of the single robot experiments, we compare the performance of Deterministic Actuated Sensing, Adaptive Sensing, and Multiscale Sensing. We did not include the results for static sensing because the results are analogous to Deterministic Actuated Sensing in terms of the number of observations made. Additionally, for the light intensity application static sensors make observations almost instantaneously; therefore, the time required for static sensors to make all observations is insignificant. This is because a PAR (Photosynthetically Active Radiation) sensor, which is used to make the observations, has a measurement time of only 30 \( \mu s \). To compare the performance of environmental monitoring algorithms for the application of light intensity, we first extracted the regions of interest, i.e. the sunlit bright regions, from the images. Fig. (1b) illustrates the bright regions in the 10 a.m. image. The root mean square error (RMSE) of the bright regions was found for each image when no observations were made. This is equivalent to an RMSE of 100%. We then found the value for 5%, 10%, 20%, 30%, and 40% RMSE.

When an observation is made, we average the pixel intensities in the 3x3 pixel neighborhood of each location to reduce
noise and emulate the observation made at the corresponding location. Given a set of observations, we generate an estimate of the field using triangle-based cubic interpolation; although, any number of spatial predictors also could have been used. For each algorithm, we find the minimum number of observations required to produce a field estimate that achieves a given percent RMSE and the respective run time based on the observation density and the measurement or dwell time required to make an observation. For all simulations, the measurement time was set to 30 $\mu$s, which for the light intensity application is the time it takes for a PAR (Photosynthetically Active Radiation) sensor to take a measurement. Run times were calculated based on the mobile robot speed, in $cm/s$, and the distance traveled. The distance traveled was calculated via the number of pixels visited and by using the following relationship: Each image is 768 x 480 pixels and covers an area of 6 m x 4 m; Thus, each pixel represents 0.78 cm.

A. Algorithmic Comparisons

Fig. (2) illustrates the results for the performance comparison of Deterministic Actuated Sensing (DAS), Adaptive Sensing (AS), and Multiscale Sensing (MS). All results shown are for the image taken around 10 a.m. Similar results were found for the image taken around 8 a.m., but they are not shown here due to space constraints. For Fig. (2a), Fig. (2b), and Fig. (2c), we varied the robot speed from 5 $cm/s$ to 40 $cm/s$. For these three figures the y-axis represents the run time (in seconds) to achieve 10%, 30%, and 40% root mean square error (RMSE), respectively. For all percent RMSEs shown, MS outperforms both DAS and AS. For 30% RMSE, AS begins to narrow the performance difference with DAS for algorithm run time. By 40% RMSE, AS outperforms DAS and begins to narrow the performance gap with MS. For Fig. (2a), Fig. (2b), and Fig. (2c), the performance differences between DAS, AS, and MS are significantly more pronounced for slower robot speeds than for higher robot speeds. Additionally, these results give insight into the relationship between algorithm performance and temporally dynamic phenomena. Here, we report the results on a static image. If the phenomena is temporally dynamic, then the same relationship between rate of change of the phenomena and robot speed can be maintained by adjusting the robot speed accordingly. Thus, if the phenomena is evolving at a relatively slow rate, this would be equivalent to running the robot at a slightly higher speed. Therefore, the results from the higher robot speeds would be relevant. Conversely, if the phenomena is evolving at a relatively fast rate, this would be equivalent to running the robot at a slower speed. Therefore, the results from the slower robot speeds would be relevant.

Fig. (2d) compares the run times (in seconds) required for
Next, we examined how the number of observations required to meet a given % RMSE compares across DAS, AS, and MS. On the x-axis of Fig. (2g) is the percent RMSE each algorithm is required to achieve and on the y-axis is the number of observations required to meet the given requirement. Fig. (2h) and Fig. (2i) show zoomed-in views of Fig. (2h). Note that the number of observations required is independent of the run time. For reference, the number of static sensors required for a fidelity reconstruction of 5%, 20%, and 40% RMSE is 92758, 770, and 95 sensors, respectively. MS outperforms both AS and DAS until 30% RMSE at which point MS and AS converge on 56 observations required. By 40% RMSE, AS performs slightly better than MS. AS requires only 24 observations where as MS requires 39 observations. This is because MS must capture the variations at the task boundaries well enough to predict the internal area of the task with high enough fidelity. As a result, as the % RMSE increases, MS decreases the number of observations but at a slower rate than AS.

**B. Multi-Robot Experiments**

Fig. (3) illustrates the results for Multiscale Sensing when it is implemented using multiple robots for servicing tasks, i.e., the sunlit bright regions in the case of light intensity monitoring. When the Multiscale Sensing algorithm begins, if 1 to 4 mobile robots are utilized it is assumed that one robot is available in each corner of the environment. This serves as the mobile robots’ starting locations. When 4 to 8 robots are utilized, it is assumed that two mobile robots are available in each corner of the environment and so forth. We assign the highest priority task to the nearest robot and each mobile robot services each task until completion before moving on to service the next task. Fig. (3a) and Fig. (3b) compare the run time (in seconds) to achieve a specified percent root mean square error. The three RMSEs analyzed were 10%, 20%, and
In the future, we plan to perform a more in depth performance comparison of Static Sensing, Deterministic Actuated Sensing, Adaptive Sensing, and Multiscale Sensing by analyzing a diverse set of dynamic spatio-temporal environmental phenomena.

REFERENCES

Connectivity-based Localization in Robot Networks

Tobias Jung, Mazda Ahmadi, Peter Stone
Department of Computer Sciences
The University of Texas at Austin
Email: {tjung,mazda,pstone}@cs.utexas.edu

Abstract—Consider a small team of autonomous robots, each equipped with a radio, that are deployed in an ad hoc fashion and whose goal it is to act as signal relay nodes to form a temporary, adaptive, and highly robust communication network. To perform this type of self-optimization and self-healing, relative localization (i.e. knowing direction and distance to every other robot in the network) is necessary. In a sense, the problem is similar to the one studied in ad hoc sensor networks. The key differences are that (1) anchor nodes with known locations are not available; that (2) the connectivity graph is very sparse, because of a comparatively small number of nodes involved; and that (3) the communication nodes are actually mobile robots such that apart from location we also have to estimate the directions to other nodes (which can not be obtained from a single time slice). To solve this problem, we propose a global approach that exploits the mobility of the robots to obtain multiple connectivity measurements over a small time window. Together with the odometry of individual robots, we then try to estimate underlying locations that best explain the observed connectivity data by minimizing a suitable stress function. Through simulation of a concrete real-world scenario we show that our approach performs reasonably well with as few as ten robots. We examine its performance both under outdoor and indoor conditions (i.e. uniform and non-uniform signal propagation). In addition, we also consider the case where we are able to observe the distance between connected robots, which further improves accuracy substantially.

I. INTRODUCTION

While radio communication generally works well in open (line of sight — LOS) environments, impeding features such as walls or other obstacles hinder the propagation of radio signals in indoor (non-LOS) environments such that direct communication often becomes impossible. However, having communication when operating in these environments is highly desirable for many real-world situations (for example, establishing and maintaining communication throughout disaster-stricken urban areas to coordinate rescue and emergency operations). A solution we explore in this paper is to employ a team of intelligent communication robots that are scattered through the environment and whose task it is to autonomously create and sustain a temporary communication network. To maintain reliable communication over extended periods of time, the robots will have the ability to move so that they can constantly adjust their position and always find the best signal. In particular, as the network is intended to be multi-path and multi-hop, the loss of individual nodes should not stop data from finding its way to its endpoint via alternative node paths. Thus, if a single robot fails or is destroyed, the rest of the network will have to reposition to cover-up the resulting black spot. In this context, having relative localization (i.e. knowing direction and distance to every other robot in the network is necessary, or at least very useful, for performing the desired self-optimization and self-healing behavior [1].

For this paper, we describe a history-based approach that jointly estimates relative positions of the robots by combining individual odometry with the global connectivity graph over a small time window. More specifically, we will consider the localization problem under the following constraints (which stem from our particular application):

- the robots are deployed ad-hoc, scattered throughout the environment and unaware of their initial global location and heading. The goal is to estimate their relative position in the network (i.e. distance and direction to all other robots).
- as there are no stationary beacon nodes with known locations, the relative locations of the robots can only be jointly determined from connectivity.
- the total number of robots is small, such that any technique relying on dense coverage will fail.
- the robots are mobile\(^1\), but we do not assume control over their movement. However, the robots are equipped with fairly accurate odometry sensors.
- the robots may have to operate under non-LOS conditions, such that signal strength measurements may become unreliable and cannot be used to directly infer the underlying distance.
- the robots (small, inexpensive units) lack any sensor device that would otherwise enable sophisticated mapping of the environment.

II. RELATED WORK

The localization problem for a single mobile robot has been extensively studied in the past. Mobile robot localization usually works by assuming a prior map and then trying to determine the robot’s position with respect to that map by integrating motion and sensor data over time using Bayesian filtering. More advanced algorithms try to learn a map and solve the localization problem simultaneously (SLAM). For a summary of these methods, see [10]. However, chiefly because of two reasons, these methods are not directly applicable to our situation: (1) the robots are deployed in uncharted territory and lack sensors for mapping the environment; and

\(^1\)It is a crucial assumption we make throughout this paper that the robots actually change their positions such that their connectivity changes over time. Only this way will we obtain sufficient data to solve an otherwise seriously underdetermined reconstruction problem.
experiments we consider only 10 nodes) with the average consists of a comparatively small number of nodes (in our in its neighborhood). A detailed survey of this area is provided (which usually means every node has contact only with nodes other node by virtue of having restricted communication range (i.e. WiFi access points), the goal is to estimate the objects (i.e. WiFi localization). Here, given a number of stationary simultaneously localizing multiple robots.

(2) these methods do not adequately address the problem of simultaneously localizing multiple robots.

Map-free localization is a common objective in the context of WiFi localization. Here, given a number of stationary objects (i.e. WiFi access points), the goal is to estimate the location of a mobile device just based on signal strength readings. In [4] this was done for an indoor environment by first learning signal strength maps for the individual access points from labeled ground truth data using GP regression, and subsequently using Bayesian filtering to estimate the location of the device. In [3] this approach was extended to work without labeled data, using GPLVM, a recent method for non-linear GP-based dimensionality reduction. However, in order to produce reasonable results, dense and overlapping coverage, i.e. a large number of access points, was required. Another example of applying dimensionality reduction for map-free localization is presented in [11]. There, the objective was to determine the location of a number of stationary objects, using first a mobile robot to obtain temporally related (and therefore similar) measurements of some spatial relationship (such as visibility, distance, direction). Then dimensionality reduction was applied to a history of these high-dimensional measurements to produce the low-dimensional locations of the objects.

Localization in ad-hoc sensor networks probably comes closest to what we want to achieve, being both map-free and specifically tailored to simultaneously localize multiple objects from a pairwise similarity measure (radio communication). The methods exploit the fact that each node in a communication network constrains the possible locations of every other node by virtue of having restricted communication range (which usually means every node has contact only with nodes in its neighborhood). A detailed survey of this area is provided by [2]. However, many of the existing technologies rely on large numbers of carefully placed anchor nodes whose location must be known in advance. In addition, many localization methods require more information than just connectivity, and use distance or angle measurements from the anchor nodes to apply multilateration or triangulation techniques to find coordinates of the unknown nodes [6]. A method that does not rely on anchor nodes, and works even when only connectivity is available, is MDS-MAP [8]. However, connectivity-only MDS-MAP requires dense networks with many nodes (on the order of hundreds) and a high degree of connectivity (average number of neighbors). In our case, the communication network consists of a comparatively small number of nodes (in our experiments we consider only 10 nodes) with the average number of neighbors being about 0-2; the reconstruction problem will thus be seriously underconstrained in the absence of movement. Finally, localization in sensor networks usually assumes that the communication nodes are stationary. The novelty/contribution of this paper is to explicitly consider (and exploit) the case where the nodes are in fact autonomous mobile robots and the induced connectivity graph changes over the time (as the robots move around, edges break up and new edges are formed).

III. DETAILED DESCRIPTION OF OUR APPROACH

Our approach is a history-based global approach that tries to jointly estimate relative positioning of robots: first, we combine the individual observation and motion histories of all robots in the network. We then estimate locations such that the induced connectivity graph is most consistent with the observed connectivity graph (for all time steps in the history). In the following, we will consider $N$ robots moving in a 2D world for a number of $T$ time steps.

A. Individual motion

Let $x(i,t) = x_i^t$ denote the $x$-coordinate, $y(i,t) = y_i^t$ the $y$-coordinate, and $\theta(i,t + 1) = \theta_i^{t+1}$ the heading of robot $i$ at time $t$. As illustrated in Figure 1, the coordinates are given with respect to a global coordinate system, and the heading is given with respect to the positive $x$-axis (note the use of $t+1$ to indicate heading at time $t$).

Given only the coordinates $(x_1^1, y_1^1), \ldots, (x_T^1, y_T^1)$, we can compute the associated (absolute) heading for all but the last time step. First, introduce the distance robot $i$ moved between two successive time steps along each of the coordinate directions:

$$d_x(i,t) := x(i,t) - x(i,t - 1), \quad t = 2 \ldots T$$
$$d_y(i,t) := y(i,t) - y(i,t - 1), \quad t = 2 \ldots T.$$ We then have

$$\theta(i,t) := \text{atan2}(d_y(i,t), d_x(i,t)), \quad t = 2 \ldots T,$$

which gives us the heading (for robot $i$) at time steps $t = 1 \ldots T - 1$ (see Figure 1).

Conversely, if we know the start pose $(x_1^1, y_1^1, \theta_1^2)$, i.e. location and heading at time $t = 1$, and are given both the distances robot $i$ traveled between successive time steps, i.e.

$$\delta(i,t) := (d_x(i,t)^2 + d_y(i,t)^2)^{1/2}, \quad t = 2 \ldots T,$$
and the relative changes in heading, i.e.
\[ d_\theta(i, t) := \theta(i, t) - \theta(i, t - 1), \quad t = 2 \ldots T, \]
we can compute the locations for all time steps \( t = 2 \ldots T \) beyond the first:
\begin{align*}
\theta(i, t) &= \theta(i, 2) + \sum_{k=3}^{t} d_\theta(i, k) \\
x(i, t) &= x(i, 1) + \sum_{k=2}^{t} \delta(i, k) \cos(\theta(i, k)) \\
y(i, t) &= y(i, 1) + \sum_{k=2}^{t} \delta(i, k) \sin(\theta(i, k)).
\end{align*}

Now assume we do not know the initial pose of node, but could observe the history of its motion, i.e. we could observe the \( \delta \)'s from Eq. (1) and \( d_\theta \)'s from Eq. (2). Let \((x_1^{\text{start}}, y_1^{\text{start}}, \varphi_1^{\text{start}})\) be a guess for the true unknown initial pose \((x_1^0, y_1^0, \theta_1^0)\) in a global coordinate frame. Together with the observably odometry data \( \delta \) and \( d_\theta \), every such guess then gives rise to a path \( \{(\hat{x}_i^k, \hat{y}_i^k)\}_{i=1 \ldots N}^{k=1 \ldots T} \) via Eqs. (3)-(5):
\begin{align*}
\hat{\theta}(i, t) &= \varphi_i^{\text{start}} + \sum_{k=3}^{t} d_\theta(i, k) \\
\hat{x}(i, t) &= x_i^{\text{start}} + \sum_{k=2}^{t} \delta(i, k) \cos(\hat{\theta}(i, k)) \\
\hat{y}(i, t) &= y_i^{\text{start}} + \sum_{k=2}^{t} \delta(i, k) \sin(\hat{\theta}(i, k)).
\end{align*}

### B. Joint connectivity

Whereas motion is handled independently for every individual node, the connectivity graph induced by the locations of all nodes at any time \( t \) is a global property of the whole network. Let \( g(i, j, t) \) denote the Euclidean distance between node \( i \) and node \( j \)
\[ g(i, j, t) := \left( (x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2 \right)^{1/2} \]
for \( i = 1, \ldots, N - 1, j = i + 1, \ldots, N, \ t = 1, \ldots, T. \) Furthermore, let \( c(i, j, t) \) denote a binary connectivity variable
\[ c(i, j, t) :=
\begin{cases}
1 & \text{if node } i \text{ and } j \text{ could communicate at time } t \\
0 & \text{else}.
\end{cases}
\]
Ideally, under LOS conditions, signal strength falls off uniformly and connectivity is thus directly related with the underlying distance. In this case, we can model connectivity by setting \( c(i, j, t) = 1 \) if \( g(i, j, t) \leq R_{\text{max}} \), where \( R_{\text{max}} \) is the maximum communication distance. Under non-LOS conditions the situation is substantially more difficult; in general, signal strength, and thus connectivity, is then less strongly correlated with distance (e.g. two nodes could be physically close but separated by a wall that absorbs the signal). In our simulations (see Section 4), we will use a sophisticated ray-casting approach to model physical signal propagation in an indoor environment. However, since our nodes are incapable of determining the topology of their environment (and thus are unaware of walls), we will model the uncertainty using a simple probabilistic relationship between pairwise distance and connectivity.

### C. Objective

Regardless of what underlying physical process gave rise to the measured signal strength, assume we could observe the joint connectivity of the network together with the individual motion of nodes over a history of \( T \) time steps. Figure 2 depicts this situation. Our goal is now to collectively find initial poses for all nodes, i.e. determine vector \( \vec{x} := (x_1^{\text{start}}, y_1^{\text{start}}, \varphi_1^{\text{start}}, \ldots, x_N^{\text{start}}, y_N^{\text{start}}, \varphi_N^{\text{start}}) \in \mathbb{R}^{3N}, \) such that when we apply the odometry to expand the paths over time, the connectivity graph induced by the underlying estimated locations best agrees with the observed connectivity graph induced by the unknown true locations. Let \( \tilde{g}_\varphi(i, j, t) \) denote the distance between estimated locations for \( i \) and \( j \) at time \( t \) for a particular \( \vec{x} \) from Eq. (9):
\[ \tilde{g}_\varphi(i, j, t) := \left( (\hat{x}_i^t - \hat{x}_j^t)^2 + (\hat{y}_i^t - \hat{y}_j^t)^2 \right)^{1/2}, \]
where \( \{(\hat{x}_i^k, \hat{y}_i^k)\} \) is obtained via Eqs. (6)-(8).

In the following we will consider two different scenarios: in the first scenario, we assume, as described above, that we can only observe whether or not two nodes were connected at any given time in the history. In a second scenario, we consider the case where range information is available for nodes that can communicate (for example, by assuming that the nodes are equipped with time difference of arrival (TDoA) hardware).

In both scenarios, to determine the best fit between connectivity graphs derived from a particular guess \( \vec{x} \) and the observed (true) connectivity, we will first define a suitable error function and then minimize it with respect to the vector of initial poses \( \vec{x} \).

Doing this for the second scenario is pretty straightforward: if we have access to the pairwise distances \( g \) for connected nodes, we can directly consider the error function
\[ E(\vec{x}) = \frac{1}{2} \sum_{t=1}^{T} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[ \tilde{g}_\varphi(i, j, t) - g(i, j, t) \right]^2 c(i, j, t), \]
where \( \tilde{g}_\varphi \) denotes the pairwise distance for locations expanded from \( \vec{x} \), and minimize it with respect to \( \vec{x} \). Note that this approach only makes use of information for nodes that are connected, i.e. ignores the nodes that are not connected. Still, as we will see in Section 4, this will give us very good results, since we are integrating information over multiple time steps.

The first scenario is more challenging because we have fewer and less reliable information from which we can infer locations. Here we assume a simple (topology-free) probabilistic model for observing connectivity between nodes at any given time given underlying locations, and find the maximum likelihood solution for \( \vec{x} \) over all the full history of
observations. As observation model we utilise a Gaussian cdf
\[ p(c_{ij}^{t} = 1 | x_{i}, t, y_{j}, t) := 1 - \Phi_{\mu, \sigma^2}(\hat{d}_{x}(i, j, t)) \]
\[ p(c_{ij}^{t} = 0 | x_{i}, t, y_{j}, t) := \Phi_{\mu, \sigma^2}(\hat{d}_{x}(i, j, t)) \]
incorporating our belief that the probability of i and j being connected will be high when i and j are close. Here, as usual, \( \Phi_{\mu, \sigma^2}(\cdot) \) is defined as
\[ \Phi_{\mu, \sigma^2}(\cdot) = \frac{1}{2} \left( 1 + \text{erf} \left( \frac{\cdot - \mu}{\sqrt{2}\sigma} \right) \right) \]
with parameters \( \mu \) being the mean, and \( \sigma^2 \) being the variance of the underlying Gaussian density. We then minimize the error function
\[ E(\hat{x}) = \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left\{ c_{ij}^{t} \log \left( 1 - \Phi_{\mu, \sigma^2}(\hat{d}_{x}(i, j, t)) \right) + (1 - c_{ij}^{t}) \log \left( \Phi_{\mu, \sigma^2}(\hat{d}_{x}(i, j, t)) \right) \right\} \]
which is the associated negative log-likelihood. The first term in the sum corresponds to the connected nodes, and is large when two nodes were observed as being connected, but their estimated locations are far apart. Conversely, the second term corresponds to the non-connected nodes, and is large when two nodes were observed as being not connected, but their estimated locations are close. Admittedly, this is a rather broad approach that completely ignores the underlying topology, but it is the best we can do given the little information we have.

IV. EXPERIMENTAL RESULTS

As test environment we simulate an indoor office space, a small sketch of which is shown in Figure 2b. We use CYBELEPRO, a proprietary simulator [X], to model the motion of robots in the environment under real-world conditions. For the following experiments we consider a team of \( N = 10 \) robots. At the start of the simulation the robots were artificially placed such that they were well spread out and located in different parts of the environment. Under this setup, the nodes did not form a fully connected network initially (some nodes were deliberately placed outside the communication range of all other nodes), but were able to establish a fully connected mesh after performing some exploratory movement. This setup simulates the initial phase of our intended application: scattered nodes explore their immediate surroundings and search for other nodes until a fully connected network is established.

We consider two scenarios for the propagation of radio signals. The first one corresponds to an open environment and models received signal strength with radial symmetric attenuation: two nodes were allowed to communicate when their distance was below 9m, irrespective of impeding walls and other obstacles. In the second scenario we actively investigate indoor localization, where the underlying topology (see Figure 2b) strongly effects the propagation of the signal. This was implemented by precomputing a signal strength map for the whole office area: first, the space was evenly divided into a grid of cells. Then the received signal strength between every pair of cells was calculated by modeling the signal propagation and attenuation by integrating the effect of obstacles along a ray cast between the cells [9]. Note that for the purpose of localization the robots themselves were unaware of underlying topology.

A. Range-based localization

For each of these two scenarios, our primary interest is in obtaining relative localization using distance-based measurements. We therefore assume that whenever two robots can communicate, we can observe the underlying distance. To solve the localization problem, we collected odometry of every individual robot, together with joint connectivity data and associated distances, over a time horizon of 60 seconds\(^2\), giving us \( d(i, t), \delta(i, t), c(i, j, t), p(i, j, t) \). Assembling the data across the nodes and time steps can be done by having every node broadcast its local odometry and connectivity data (at this point we do not consider the cost of communication). Since the underlying minimization problem in Eq. (10) does not admit a closed form solution, we employ the efficient and Hessian-free scaled conjugate gradients algorithm [5] as

\(^2\)The simulation is updated once every 100ms. To reduce communication, we only considered measurements at the rate of 1Hz. We subsampled the data and adjusted the odometry correspondingly.
an iterative gradient-based solver. The computation of the gradient is described in Appendix A. Since the problem is nonconvex, we have to deal with multiple local minima. To ensure a high-quality solution, we ran the solver multiple times, each time restarting with a random initial value for $\hat{x}$. In our case we used 100 restarts and ran the optimizer for a maximum of 200 iterations. Note that in principle, because all the nodes in the network share the same data, this computation can be carried out in a distributed way.

Figure 3a shows the result for both scenarios using the same data. Since our goal is relative localization, we determine the quality of the reconstruction by taking the angular error between any two robots (i.e. the angle under which node $i$ sees node $j$ with respect to its own heading) and their relative distance. The plot shows the corresponding errors averaged over all pairs of robots and time steps (errorbars are given in one standard deviation). In addition, we also consider the effect of making noisy measurements: all of the observations (i.e. both odometry terms and the range) were corrupted by white noise with standard deviation of 1% (denoted as 'low') or 10% (denoted as 'medium').

B. Connectivity-based localization

In a second series of experiments, we repeated all of the above, this time assuming localization from connectivity alone. In this case, the relative locations were estimated by solving Eq. (11). The computation of the gradient is given in Appendix B. As we can see from the results in Figure 3b, the reconstruction error for both the outdoor and the indoor scenario is, unsurprisingly, substantially higher for connectivity-based localization than it is for range-based localization: in the outdoor case the angular error on the average is about 2-4 degrees, whereas with range-based localization it was less than 1 degree (with zero noise). For the case of 'low noise', Figure 4 shows in more detail how the error is distributed over its values. To give a visual impression of the quality of reconstruction, Figure 5 shows both the estimated and true locations, after fitting the relative coordinates to the true coordinates via Procrustes analysis [7]. Still, as most applications only require approximate localization, an error of this order of magnitude will usually be acceptable, considering

that we can achieve this result from connectivity alone. Also note that our method for connectivity-based localization is far less susceptible to noise. On the other hand, under indoor conditions range-based localization still continues to work with negligible performance loss, whereas connectivity-based localization breaks down and produces an angular error of about 20 degrees on the average.
APPENDIX A: GRADIENT FOR RANGE-BASED

The gradient of \( \Phi^{\text{start}}(\nu) \) for the \( \nu \)-th component of \( \bar{x} \), where \( x^{(\nu)} \) is one of \( x^{\text{start}}, y^{\text{start}}, \phi^{\text{start}} \) for \( \nu = 1 \ldots N \), is obtained from

\[
\begin{align*}
\partial_{\nu}(\nu) \Phi = & \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left[ \hat{\phi}(i, j, t) - \rho(i, j, t) \right] c(i, j, t) \\
& \cdot \left[ \partial_{\nu}(\nu) \hat{\phi}(i, j, t) \right].
\end{align*}
\]

(12)

Since only those terms in the sum where either index \( i \) or index \( j \) is equal to \( \nu \) are unequal from zero, we can rewrite the summation in the following way:

\[
\begin{align*}
\partial_{\nu}(\nu) \Phi = & \sum_{t=1}^{T} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left[ \hat{\phi}(i, j, t) - \rho(i, j, t) \right] c(i, j, t) \\
& \cdot \left[ \partial_{\nu}(\nu) \hat{\phi}(i, j, t) \right] + \\
& \sum_{i=\nu+1}^{N} \left[ \hat{\phi}(i, \nu, t) - \rho(i, \nu, t) \right] c(i, \nu, t) \\
& \cdot \left[ \partial_{\nu}(\nu) \hat{\phi}(i, \nu, t) \right].
\end{align*}
\]

(13)

To compute the derivative of the pairwise distances \( \hat{\phi} \), we have to consider the following 6 different cases. First, consider the derivative with respect to the initial \( x \)-coordinate of the \( \nu \)-th robot. Repeated application of the chain rule gives:

\[
\partial_{\nu} x^{\text{start}} \hat{\phi}(i, \nu, t) = - \left[ \hat{\phi}(i, \nu, t) \right]^{-1} d_{\nu}(i, \nu, t),
\]

(14)

where \( \hat{d}(i, \nu, t) := \hat{x}(i, t) - \hat{x}(\nu, t) \). Second, doing the same for the \( y \)-coordinate, we obtain:

\[
\partial_{\nu} y^{\text{start}} \hat{\phi}(i, \nu, t) = - \left[ \hat{\phi}(i, \nu, t) \right]^{-1} \hat{d}_{\nu}(i, \nu, t),
\]

(15)

where \( \hat{d}_{\nu}(i, \nu, t) := \hat{y}(i, t) - \hat{y}(\nu, t) \). Third, for the derivative with respect to the initial heading angle \( \phi^{\text{start}} \), we obtain (repeated application of the chain rule):

\[
\begin{align*}
\partial_{\nu} \phi^{\text{start}} \hat{\phi}(i, \nu, t) = & \left[ \hat{\phi}(i, \nu, t) \right]^{-1} \\
& \cdot \left[ \hat{d}(i, \nu, t) \hat{\phi}(\nu, t) - \hat{d}_{\nu}(i, \nu, t) \hat{\phi}(\nu, t) \right],
\end{align*}
\]

(16)

where

\[
\begin{align*}
\hat{\alpha}(\nu, k) := & \sum_{k=2}^{t} \delta(k, \nu, k) \cos \left( \phi^{\text{start}} + \sum_{l=2}^{k} \phi(l, \nu, l) \right) \\
\hat{\beta}(\nu, k) := & \sum_{k=2}^{t} \delta(k, \nu, k) \sin \left( \phi^{\text{start}} + \sum_{l=2}^{k} \phi(l, \nu, l) \right).
\end{align*}
\]

The remaining three cases correspond to the derivatives with respect to the first index in \( \hat{\phi}(\nu, i, t) \), here we just have to invert the sign of Eqs. (14)-(16). Note that for the first time step \( t = 1 \) the derivative with respect to \( \phi^{\text{start}} \) is zero, i.e.

\[
\partial_{\nu} \phi^{\text{start}} \hat{\phi}(\nu, i, 1) = \partial_{\nu} x^{\text{start}} \hat{\phi}(i, \nu, 1) = 0.
\]

APPENDIX B: GRADIENT FOR CONNECTIVITY-BASED

Likewise, to compute the gradient of Eq. (11), we start from

\[
\begin{align*}
\partial_{\nu}(\nu) \Phi = & \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} \Phi_{\mu, \sigma} \left( \hat{\phi}(i, j, t) \right) \left[ \partial_{\nu}(\nu) \hat{\phi}(i, j, t) \right] \\
& \cdot \left( 1 - c(i, j, t) \right) - c(i, j, t) \Phi_{\mu, \sigma} \left( \hat{\phi}(i, j, t) \right).
\end{align*}
\]

where \( \Phi' \) is derivative of the Gaussian cdf, i.e.

\[
\Phi'_{\mu, \sigma^2} (\cdot) = \exp \left\{ - \frac{(\cdot - \mu)^2}{2\sigma^2} \right\} \cdot \frac{1}{\sqrt{2\pi}\sigma^2}.
\]

Rewriting this sum similar to Eq. (13), we only need to plug in the previously computed partial derivatives from Eqs. (14)-(16).

ACKNOWLEDGMENTS

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CNS-0615104), DARPA (FA8750-05-2-0283 and FA8650-08-C-7812), the Federal Highway Administration (DTFH61-07-H-00030), and General Motors. This particular research was enabled by cooperation and technical support from Vikram Manikonda and his team at IAI.

REFERENCES

Utilizing Sensor-Based Robotic Terrain Coverage
Meral Camci and Sanem Sariel-Talay, Member, IEEE

Abstract—This paper presents Enhanced - Least Recently Visited (ELRV) approach, an improved version of an existing sensor-based robotic terrain coverage algorithm, Least Recently Visited (LRV). LRV and ELRV are both proposed for either a single robot or a multi-robot team to cover an unstructured environment by incremental deployment of sensor nodes. LRV is used to make decisions on movement directions towards uncovered areas by means of the deployed sensors and their suggestions. Using sensors for coverage eliminates the need for mapping the environment. Unknown structure of the environment may sometimes result in inefficient coverage of the terrain as in the case of LRV. ELRV extends LRV by considering obstacle situations in unstructured environments. Furthermore, it suggests additional direction updates on sensor nodes to utilize coverage tasks. These supplementary direction updates are based on obstacle situations and decrease the total number of future direction updates when compared to LRV. Empirical evaluations of ELRV in the Player/Stage simulator present improved performance in terms of time to complete the coverage mission compared to LRV.

Index Terms—Sensor based coverage, dynamic coverage, robot sensor interaction, mobile robots.

I. INTRODUCTION

EFFICIENCY in terrain coverage is an important concern for both robot applications (e.g., cleaning, mine-removal and search and rescue operations) and sensor-based environmental monitoring applications.

This study focuses on sensor-based robotic terrain coverage and proposes Enhanced - Least Recently Visited (ELRV) as an improvement on an existing algorithm, Least Recently Visited (LRV) [1].

In LRV, robot deploys sensor nodes incrementally by using the information, on visited areas of the environment, provided by previously deployed nodes in an unknown environment. ELRV extends LRV by enhanced updating strategies for sensor nodes in initial deployment cases and for obstructed parts of the environment. These improvements help reducing overall time to complete the coverage mission.

II. RELATED WORK

Choset [2] presents an extensive survey for coverage methods. Coverage algorithms are mainly classified into two groups namely online and offline coverage approaches [2]. In offline coverage [3], [4], [5], the map of the environment is available to the robot performing the coverage task. On the other hand, in online coverage [6], [1], [7], the map or the environment is not known by the robot.

Another important criterion considered for coverage algorithms [2] is the subject of the coverage task which is a robot itself or a robot interacting with sensors positioned (deployed) statically or deployed dynamically to the environment. In robot coverage [3], [4], [7], [5] the coverage mission is performed only by the robot and the information used in coverage is gained from only the robots’ hardware (actuators and sensors of the robot). On the other hand, in sensor based coverage [6], [1] the robot interacts with sensors for completing coverage task. In sensor based coverage, sensors are either used in robotic coverage of an environment or they are directly used to form a network of sensors to monitor a terrain. Different deployment strategies may be performed for these applications. [6] uses static coverage in which sensors are positioned statically to cover every part of the environment with their sensor shadows at every time. This method is efficient for known environments (like offline coverage described in [2]). In LRV [1], as an online coverage strategy, a robot covers the environment step by step without having a map or plan. In this method the robot deploys sensor nodes incrementally according to the information obtained from the previous deployment status which is called dynamic coverage [1].

The number of robots performing the coverage task is another criterion in classifying the coverage methods. Coverage is performed by a single robot [1] or multiple robots [3], [6], [4], [5], [7]. Single robot coverage of an environment takes longer time when compared to multiple robot coverage. If robot failures occur, the coverage mission is performed by other robots in the team.

This paper focuses on an online strategy for a single robot to cover an unknown environment with the suggestions of the sensor nodes deployed incrementally. Particularly, LRV algorithm is investigated and some improvements are suggested as a new strategy on top of the main mechanisms of it. LRV has some deficiencies in deploying sensor nodes near obstacle locations and in some update strategies for newly deployed sensors. The main motivation of this research is to enhance LRV to eliminate these deficiencies.

III. ENHANCED-LRV

LRV [1] makes use of sensor nodes for storing coverage information which is used by a robot to cover an unknown environment. Nodes are incrementally deployed by the robot based on some criteria and the movement directions of the robot are managed by the deployed nodes. During runtime, the robot deploys a new node whenever there is no node in its vicinity. This is the case for the initial deployment of the robot. There may be multiple nodes locating in its line-of-sight. In this case, it interacts with the node having highest signal strength. Throughout the paper, the current node represents
the node that the robot interacts with and the closest node represents the node with the highest signal strength after losing line of sight with the current node. Having the highest signal strength does not necessarily induce becoming the current node. As an example, a node having smaller signal strength may be selected as the current node since the candidate node (closest node) having the largest signal strength is occluded by obstacles. ELRV ensures this property by considering obstacle situations around the robot.

LRV algorithm is composed of three main parts. First part of the algorithm deals with the cases in which the robot is out of range of the current node. In this situation, the robot exploring/covering the environment deploys a new node. If there is a closest node in line-of-sight, it updates the weights of it by sending a message and moves according to this node’s suggestions. In the meantime, this node becomes the current node. The second part of the algorithm is used for the cases in which the robot discovers obstacles on its way. According to the size of the obstacles, the robot either avoids the obstacle (if it is small) or it deploys a new node (if the obstacle is a large one and there are no other nodes except from the current node in the vicinity of robot) which then becomes the current node. The number of deployed nodes around large obstacles would be higher than that of obstacle-free areas due to this policy. This is a reasonable strategy to effectively cover the environment. The last part of the algorithm is used to forward the robot to the current direction if there are no obstacles on its way. During the traversal of the nodes, weight updates for directions are performed. These weights represent the number of traversals through the corresponding directions.

An illustration of the node deployment for LRV in obstacle-free areas is given in Fig. 1 (a-d). In Fig. 1(a), the robot is deployed in the environment and starts the coverage mission. In Fig. 1(b), the robot deploys the first node in the environment since there is not a current node. The node numbered with id 1 becomes the current node at this time, and the robot uses this information and decides on the next direction towards left side. Whenever the robot is out of communication range with the first node (refer to Fig. 1 (c)), it deploys the node 2 (refer to Fig. 1 (d)).

ELRV preserves the main parts of the LRV algorithm. It differs from LRV by its node deployment strategy in obstructed parts and its weight updating strategy.

ELRV enhances LRV in its node deployment strategy for...
obstructed parts of the environment. The robot takes into consideration of the status of the environment during node deployment in ELRV. This strategy utilizes coverage in unstructured environments. The main procedures of ELRV are given in Algorithm 1.

In ELRV, weight updates for the directions are determined based on the obstacle status of the environment by the robot at the deployment time. The robot deploys a node to an obstacle-free direction (after avoiding the obstacle) and moves in that direction. There are two weight updates for the traversed and new directions that have been passed through by the robot when deploying that node. After deploying a node which becomes the current node, the robot sends two update messages to this node to declare that these two directions are traversed while deploying that node. When deploying that node, an obstacle-free direction is found. For that reason, there is no need to check if there is an obstacle in the deployed node’s suggested direction and send messages to the related node for weight updates to find an obstacle-free direction in next steps (when the robot is in the range of the closest node) the robot is in the communication range of that node again.

In LRV, the node is deployed only considering the suggested direction information. In a new node deployment case, the traversed direction is not considered and the weight updates of these directions (suggested and traversed directions) during deployment are neglected. Also the suggested direction is not determined according to the obstacle situations. This result in additional direction updates to find an obstacle-free direction later when the robot is in the range of those related nodes.

Another enhancement made in ELRV during node deployment is updating the states of nodes according to the obstacle situation of the environment. ELRV updates the traversed and suggested directions as previously mentioned. The other directions except these traversed and suggested directions may be pointing to obstructed areas. In these situations, the robot sends related update messages for these obstructed directions to the deployed nodes. The messages are related to the obstructed-directions around the related deployed nodes. This strategy reduces the communication overhead and the total coverage time when compared to LRV. LRV ignores these obstacle situations and the robot can be directed to an obstructed direction by the corresponding node. This leads the robot to an untraversable direction and results in redundant efforts to avoid obstacles. In ELRV, after node deployment, the obstructed direction(s) are detected by the robot and the immediate update message(s) for these direction(s) are sent to the deployed node. The deployed node takes these update message(s) for the obstructed direction(s), and updates the weights of these directions with a high value in order to prevent suggesting and forwarding the robot to these directions.

In ELRV, the traversed direction of the closest node is updated instead of the opposite direction of the closest direction of the closest node as in the case of LRV.

Node loop update procedure of LRV is similar in ELRV except updating the obstructed direction at the time the node is deployed. This obstructed direction is updated with a high value for preventing the node to suggest this direction to robot.

In ELRV obstructed direction(s) and the traversed directions are updated in node deployment step. This results in immediate updates of the weights which are used later and reduces the communication overhead and the total coverage time. To clarify, a sample situation is illustrated in Fig. 2. Fig. 2 (a) outlines ELRV node deployment strategy. In this scenario, the robot starts exploring the environment from the left bottom corner and the nodes are deployed in counterclockwise order. The weight updates of the node 7, is given. Because the upper side of node 7 is obstructed, the corresponding direction is updated with high value for preventing that node to suggest this direction later. In this figure the traversed direction updates can also be seen. As opposed to LRV, ELRV updates the traversed directions at the time of deployment. Fig. 2 (b) illustrates the weight update strategy of LRV for the same scenario. In LRV, the obstructed direction is not considered and the next suggested direction according to the cross order rule [1] is selected as the upper direction. This yields additional direction updates to be performed later by the robot.

IV. IMPLEMENTATION DETAILS

The Pioneer 2DX mobile robot equipped with 16 sonar range sensors positioned with 22.5 degrees interval is used in the simulations. Since the operation of the robot is managed by the sensors, obstacle detection and avoidance capability is considered for the robot. The sonar range finder sensors are
used for the robot to detect obstacles in its vicinity. In Fig. 3 the directions of the sonar range sensors are shown with the direction of the robot. The sonar readings are also used to determine the sizes of the obstacles to deploy new nodes in large obstacle situations. If all of these five sensor readings are in a range that is used for detecting large obstacles, according to algorithm LRV and ELRV a new node is deployed if there is no other nodes except the current node in the vicinity. All of the five sonar readings should be in the range drawn red in Fig. 3 to treat an obstacle as large.

The same obstacle avoidance and large obstacle detecting policies are applied in both algorithms to compare the results because there is no clear explanation of the obstacle avoidance policy in LRV. Also the same weight update strategy explained in [1] is used in both LRV and ELRV for the nodes managing robot to perform coverage mission.

V. SIMULATIONS AND EXPERIMENTAL RESULTS

ELRV is evaluated in Player/Stage [8], [9] simulator with a simulated Pioneer 2DX mobile robot having a sonar range finder.

Comparison criteria are selected as the percentage of the covered areas, time to complete the coverage mission and the number of nodes deployed. Total coverage result is obtained by calculating the number of uncovered points by the following intuitive formula:

\[
\text{Coverage\%} = \frac{(\text{Area} - \text{Uncovered Area})}{\text{Area}} \quad (1)
\]

The simulations are ended when the total coverage is full for obstacle-free environments and minimum 98% coverage is obtained for environments with obstacles.

Experiments are designed in three sets for different environments with different obstacle densities. The performance of the algorithms are evaluated in empty environments, in 5 different environments with 10% obstacle densities and 20% obstacle densities in the first, second and the third set respectively.

The results of the first set of experiments are given as the average of 5 different runs in Table I.

The total coverage time, the coverage percentage and the number of nodes (given as a range) deployed are presented in the first, second and the third columns respectively. As the results illustrate, although the number of nodes deployed in the environment is similar in both LRV and ELRV, ELRV outperforms LRV in terms of time to complete the coverage mission.

The results of the second set of experiments are given as the averages of 5 different runs for each of the 5 different environments with 10% obstacle densities (Table II). The results of the third set of experiments are given as the averages of 5 different runs for each of the 5 different environments with 20% obstacle densities (Table III).

The presented results reveal that there is no significant difference in the number of deployed nodes comparing the two algorithms. The main difference of the results lies in the total coverage time of the environments which are either obstacle-free or obstructed. ELRV improves coverage efficiency approximately 33% in terms of time to cover the environment compared to LRV. This efficiency improvement is ensured by the node deployment strategy (obstructed directions update) and the traversed direction weight updates of nodes performed in ELRV. The new node deployment strategy considering the near obstacle information and updating weights accordingly prevents the robot from getting stuck into infinite loops. This strategy prevents a further forwarding to an obstacle location and accordingly extending the coverage time. According to the new node deployment strategy, additional direction updates for the traversed direction of the robot and the suggested direction of the node are performed. The robot uses the information gained when deploying a node without waiting any further steps.

It can also be observed from the experimental results that there is not a direct relation between the obstacle percentage of the environment and the coverage time.
### Table II

<table>
<thead>
<tr>
<th>ENV</th>
<th>Time (min, µ)</th>
<th>Coverage (%)</th>
<th># of nodes (range)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Env1</td>
<td>27.96</td>
<td>99.71</td>
<td>23-24</td>
</tr>
<tr>
<td>Env2</td>
<td>17.73</td>
<td>98.52</td>
<td>24-25</td>
</tr>
<tr>
<td>Env3</td>
<td>12.63</td>
<td>99.97</td>
<td>24-25</td>
</tr>
<tr>
<td>Env4</td>
<td>31.75</td>
<td>99.39</td>
<td>25</td>
</tr>
<tr>
<td>Env5</td>
<td>22.21</td>
<td>99.76</td>
<td>24-26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LRV</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Env1</td>
<td>89.58</td>
<td>98.23</td>
<td>25-26</td>
</tr>
<tr>
<td>Env2</td>
<td>50.68</td>
<td>99.53</td>
<td>24-26</td>
</tr>
<tr>
<td>Env3</td>
<td>34.33</td>
<td>98.64</td>
<td>25-26</td>
</tr>
<tr>
<td>Env4</td>
<td>93.75</td>
<td>99.14</td>
<td>26</td>
</tr>
<tr>
<td>Env5</td>
<td>69.93</td>
<td>99.25</td>
<td>24-25</td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th>ENV</th>
<th>Time (min, µ)</th>
<th>Coverage (%)</th>
<th># of nodes (range)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Env6</td>
<td>11.88</td>
<td>99.36</td>
<td>23-25</td>
</tr>
<tr>
<td>Env7</td>
<td>26.50</td>
<td>99.11</td>
<td>25</td>
</tr>
<tr>
<td>Env8</td>
<td>21.35</td>
<td>98.62</td>
<td>24-25</td>
</tr>
<tr>
<td>Env9</td>
<td>30.20</td>
<td>99.71</td>
<td>26</td>
</tr>
<tr>
<td>Env10</td>
<td>28.61</td>
<td>99.34</td>
<td>25-26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LRV</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Env6</td>
<td>46.05</td>
<td>98.31</td>
<td>26</td>
</tr>
<tr>
<td>Env7</td>
<td>63.36</td>
<td>98.43</td>
<td>27</td>
</tr>
<tr>
<td>Env8</td>
<td>56.80</td>
<td>98.26</td>
<td>25-27</td>
</tr>
<tr>
<td>Env9</td>
<td>77.40</td>
<td>99.84</td>
<td>25-26</td>
</tr>
<tr>
<td>Env10</td>
<td>74.20</td>
<td>99.13</td>
<td>25</td>
</tr>
</tbody>
</table>

A snapshot of a simulation run for ELRV in an obstacle-free environment is given in Fig. 4. LRV performs nearly the same placement of nodes but with a longer coverage time as previously mentioned in Table I.

Fig. 5 and Fig. 6 illustrates sample snapshots from the simulation runs in an environment 20% obstacle density for the algorithms ELRV and LRV respectively. A final remark illustrated in Fig. 5 is 100% coverage cannot always be achieved in obstructed environments because of the uncovered places remained in large obstacle regions (e.g., the uncovered region in Fig. 5 is painted with green in the figure).

**VI. Conclusion**

This paper presents ELRV as an enhanced extension of LRV. Unknown structure of the environment may sometimes result in inefficient coverage of the terrain as in the case of LRV. ELRV extends LRV by considering obstacle situations...
and suggesting additional weight updates on sensor nodes to utilize the coverage mission. These supplementary updates ensure efficiency in coverage. Empirical evaluations of ELRV in the Player/Stage simulator present significant improvements provided by ELRV in reducing the coverage time compared to LRV. Analyzing the practical limitations of the proposed scheme and connectivity analysis of the nodes is left as a near-future work. The future work also includes integration of sensor failure recovery mechanisms to the design of ELRV.

Although the performance of ELRV is presented in a realistic simulator, porting ELRV on real robots and an extensive set of experiments for real environments are planned for analyzing the performance of ELRV to a greater extent.

REFERENCES


Cooperative Target Tracking with a Communication Limited Active Sensor Network

Maciej Stachura*, Anthony Carfang*, and Eric W. Frew*

*Aerospace Engineering Sciences
University of Colorado, Boulder, CO 80309

Abstract—This paper presents a framework to combine sensing and communication in an active sensor network. The communication is modelled as a packet based erasure channel where the probability of a successful transmission is based on empirical data. An active sensor network with two mobile sensors and a fixed ground station tracking a moving target with unknown dynamics is simulated to demonstrate the algorithm. For this scenario, it is shown that factoring the communication into the planning will reduce the estimation error.

I. INTRODUCTION

Active sensor networks have grown in popularity in recent years, as have applications and research for unmanned aerial systems (UAS). The mobile nature of UAS have a distinct advantage in estimation problems over stationary sensor networks. With any active sensor network, limitation on communication range is an inherent concern to sensor placement. In a mobile sensor network such as UAS, this issue results in restricted movement as well. It is therefore not enough to compute sensor trajectories that are optimal only from an estimation standpoint. The limitations of the communication systems must be factored in when commanding the mobile sensors. There are many applications for such networks including target search and tracking, surveillance, or the gathering of severe storm data.

Much research has addressed this problem using a variety of approaches. In [1], Mostofi et al. examined optimal sensor placement in the presence of imperfect communication by minimizing a cost function involving both sensor and communication noise. Logothetis et al. presented an information theoretic approach to observer path design in bearings-only tracking [2]. In [3], the optimal sensor location is found by computing the mutual information from a particle filter, enabling the possibility for non-Gaussian distributions. However, the focus of both [2] and [3] is on the sensor placement without considering the effects of communication. In [4] two approaches are taken to optimally place the sensors. The first is to minimize the communication cost, which is based on the expected number of re-transmissions, subject to a lower bound constraint on the sensing quality. The second method employed is the opposite, in that the sensing quality is maximized with an upper bound on the communication cost.

This paper differs from others in that the probability of successfully transmitting a packet containing a measurement is factored into the computation of the expected amount of information to be received per packet. The expected information metric is based on the Extended Information Filter formulation. The sensor network is configured to maximize this expected information gain in order to optimize the data collection.

The rest of the paper is organized as follows: section 2 contains the communication model and bandwidth analysis, section 3 describes Expected Information metric along with the optimization, section 4 contains the system models, section 5 is the simulation and results, and section 6 concludes the paper.

II. COMMUNICATION MODEL

A. Modelling the RF Environment

For the active sensor network considered, an ad-hoc wireless network provides the sensor-to-base-station communication. Radio frequency (RF) propagation depends on many unknown parameters beyond relative position such as multi-path, environmental fading, noise, and interference. In particular, an RF communication link can be characterized by the signal-to-noise ratio (SnR) defined as

\[ S_{sb}(p_s, p_b) = \frac{P_{sb}(p_s, p_b)}{N(p_b)} \]  

where \( P_{sb}(p_s, p_b) \) is the power received by the base station at position \( p_b \in \mathbb{R}^2 \) from the transmission of the UA located at \( p_s \in \mathbb{R}^2 \). Radio propagation is specified using the standard empirical radio propagation model

\[ P_{sb}(p_s, p_b) = \frac{P_0 d_0}{|p_s - p_b|^\alpha} + v \]  

where \( P_0 \) is the reference power, \( d_0 \) is the reference distance, \( 2 \leq \alpha \leq 6 \) is the propagation decay exponent which is assumed constant for a given environment, and \( v \) is a randomly-distributed noise factor representing radio variability which is ignored here. \( N(p_b) \) is the environmental noise seen by the base station, and is dependent upon the quality of the receiver electronics, the noise temperature of the system, and the local RF environment of the node.

In wireless communication, broadcast energy is received by all nodes within some environment and the notion of a link is not well defined. Furthermore, many digital wireless interfaces such as IEEE 802.11 have multiple data rates [5]. When communicators are close the communication rate is high (e.g. 54 Mbps in 802.11g). When communicators are far the communication rate is lower (802.11g has 14 different
rates between 54 and 1 Mbps). The rate decision is based on the signal power to noise power ratio and packet success measurements made over the channel. This phenomenon is an example of a general principle - the well known Shannon-Hartley Law [6] which provides the theoretical maximum rate at which information passes error free over a channel.

B. Bandwidth Analysis

For large numbers of sensors in the network, the RF environment becomes more saturated simply by the collection of more data to be transmitted to the base station. With a limit on bandwidth, there is concern about the placement of the sensors, the timing with which they send data to the base station, and the route the data traverses. Evaluating the worst case scenario provides an upper bound on this concern.

In an ad hoc sensor network, the least efficient sensor configuration is a chain of sensors in a multi-hop mode. In this case, only one sensor is within range of the base station, and remaining sensors must send data to the base station by routing through progressively closer sensors in this chain. For each round of sensing, the number of hops required to get all data to the base station is

\[ H = \sum_{i=1}^{N} i = \frac{N(N - 1)}{2} \]  

where \( N \) is the number of sensor nodes in the chain. The probability that a packet for each hop will not successfully reach the destination node \( P_{\text{Fail}} \) is based on the RF environment (such as range and noise) which could cause the packet to get dropped, as well as the probability that it collides with another packet. Taking an upper bound on this probability, we assume that no node can transmit at a time in which another sensor’s data is in transit, regardless of location in the sensor chain. The probability of a collision then is then based on the transmit times of the packets and the overall time for one round of data collection.

\[ P_{\text{Fail}} = L + \frac{H S_d}{R_d} = L + \frac{H \cdot S_d \cdot R_d}{R} \]  

where \( S_d \) is the size of the data collected by the sensor, \( R_d \) is the sensing rate, \( R \) is the communication rate used, and \( L \) is the additional loss factor resulting from the RF environment. If a reliable communication protocol is used to ensure delivery, the average number of transmit attempts a sensor node will need is

\[ T = \frac{1}{1 - P_{\text{Fail}}} \]  

The time required to get all data to the base station for each round of data collection, as well as the total link usage can be calculated as

\[ \text{Time} = \frac{H \cdot T}{R} \]

\[ \text{Usage} = \frac{\text{Time}}{R_d} = \frac{H \cdot T}{R \cdot R_d} \]  

The application we are particularly interested in modeling is a scenario with 2 unmanned aircraft sensor nodes, collecting less than \( S_d = 10 \text{kb} \) of data at a collection rate of \( R_d = 2 \text{Hz} \), communicating using 802.11g at a rate of 11 Mbps. Examining the scalability of the sensor network, varying the link quality between failure rates of \( L=0 \) (perfect communication) to \( L=75\% \) shows that bandwidth is only a major concern with more than 20 sensor nodes and loss rates \( L \geq 50\% \). For our desired scenario of 2 sensor nodes in addition to the base station, bandwidth is not limiting factor. Thus the issues of communication timing and routing are negligible here. Figure 2 shows the network usage for the different scenarios described.

C. The Packet Based Erasure Model

In order to model the individual links, the binary erasure channel model [7] is used at the packet level. Each packet sent containing a sensor measurement has a probably \( \beta_{i,j} \) of reaching destination \( j \) when sent by node \( i \). Multi-hop communication will also be allowed. In the 3 node case, node \( i \) is trying to send a packet to node \( j \), and node \( k \) is available as a relay. The probability of success of the packet getting through from \( i \) to \( j \) is:

\[ \beta_{i,j} = \min \{ \beta_{i,j}, \beta_{i,k} \cdot \beta_{k,j} \} \]  

where in this case \( \beta_{i,j} \) is the probability of success of sending the pack directly to \( j \), and \( \beta_{i,k} \beta_{k,j} \) is the multi-hop probability, where the probability of each link is assumed independent. This formulation will not scale well as the number of nodes increases, however, since the proposed application is a cooperative team of Unmanned Aircraft (UA), the number of nodes will not be high.

III. OPTIMIZATION OF EXPECTED INFORMATION

A. The Expected Information

To allow the controller to plan the trajectories, the covariance (or inversely the information) must be predicted for future observations based on position. The goal is to develop an
algorithm that is independent of the estimator being used, and that depends on the current covariance from the estimator. This method is based on the Extended Information Filter [8].

Consider a discrete time nonlinear estimation problem. Let the state vector \( x_k \in \mathbb{R}^n \) evolve according to the deterministic state equation

\[
x_{k+1} = f(x_k, u_k, w_k)
\]

where \( k \) is the discrete time, \( u_k \) is the vector of deterministic inputs, and \( w_k \) is a random disturbance vector. Further, let \( z_k \) denote the measurement of the target at time \( k \) with \( x_{s,k} \) the position of the sensing UA such that

\[
z_k = h(x_k, x_{s,k}, v_k)
\]

where \( h(x_k, x_{s,k}, v_k) \) is the measurement function and \( v_k \) is zero-mean, spatially uncorrelated, white Gaussian measurement noise with covariance \( R_k \). Finally, assume some unbiased a priori statistical information about \( x_0 \) is given with estimate error covariance \( E[(x_0 - x_0)(x_0 - x_0)^T] = \Sigma_0 \), \( x_0 \) is the target state estimate at sample time \( k \). Given the initial estimate \( \hat{x}_0 \) and measurement history \( Z_k = [z_0, \cdots, z_k] \), the estimate \( \hat{x}_k \) and its covariance matrix \( \Sigma_k \) can be determined.

The information matrix \( Y_{k|k} = \Sigma_{k|k}^{-1} \) is the inverse of the estimate error covariance matrix which is given by

\[
E[Y_{k+1|k+1}] = (\Gamma_k Q_k \Gamma_k^T + \Phi_k E[Y_{k|k}]^{-1} \Phi_k^T)^{-1} + E[I_{k+1}]
\]

with

\[
Q_k = E[w_k w_k^T]
\]

\[
\Gamma_k = \nabla w f(x_k, u_k, w_k),
\]

\[
\Phi_k = \nabla x f(x_k, u_k, w_k),
\]

\[
E[I_{k+1}] = \text{expected measurement contribution to the information matrix. Note that the expectation must be used since receiving the measurements is probabilistic from the erasure channel being used to model the probability of a packet being received. Given } \beta_{i,j}, \text{ the probability of the packet successfully getting through from node } i \text{ to node } j, \text{ and noting that this is a Bernoulli distribution, the information content is:}

\[
I_{k+1} = \chi_p \cdot H_{k+1}^{-1} R_{k+1}^{-1} H_{k+1}
\]

where \( \chi_p \) is a random variable taking on either 0 or 1 from the Bernoulli distribution. For planning purposes we are interested in the expectation value of this quantity, define as:

\[
E[I_{k+1}] = \beta_{i,j} H_{k+1}^{-1} R_{k+1}^{-1} H_{k+1}
\]

with

\[
H_k = \nabla x h(x_k, x_{s,k}),
\]

An important benefit of the EIF formulation is the fact that the contributions of independent sensors sum together. Thus, it is assumed that the measurement vector is the concatenation of terms from multiple robots such that \( z_k = [z^1_k, \cdots, z^n_k] \) where \( z^n_k = h_i(x_k, x_{s,k}^i) + v^n_k \). Assuming the measurements taken by each robot are independent, the measurement contribution to the information matrix becomes

\[
E[I_{k+1}] = \sum_{i=1}^{n} \beta_{i,j} (H^i_k)^T R^{-1}_{k} H^i_k
\]

where

\[
H^i_k = \nabla x h_i(x_k, x_{s,k}^i)
\]

and in this case node \( j \) is the base node performing the data fusion.

The method discussed here allows a prediction on the covariance \( \Sigma_{k+1|k} = Y_{k+1|k}^{-1} \) at future times in order for the robot to plan its trajectory.

### B. Optimization in a Receding Horizon Framework

The cost that the controller is tasked to maximize is defined as:

\[
J_{k+T_p} = \sum_{i=1}^{T_p} \text{det}(E[Y_{k+1|k+1}])
\]

where \( T_p \) is the planning horizon, and \( E[Y_{k+1|k+1}] \) is over all sensors at time \( (k + i) \). For commanding the UA, the control input \( u^* \) is found by solving equation 20.

\[
u^* = \arg \max_{u} \{ J_{k+T_p} \}
\]

\( u^* \) is a \((N \cdot T_p \times 1)\) vector where \( N \) is the number of vehicles. The UA are then commanded for a duration, \( T_c \) (the control horizon), defined as \( T_c = \alpha \cdot T_p \), where \( \alpha \in (0, 1] \) is the control fraction.

### IV. System Models

#### A. Vehicle Model

A 2D kinematic model is employed to model the motion of the UA sensors. Let \( x_s \) denote the state of the unmanned aircraft defined as \( \chi_s = [x_s, y_s, \psi_s]^T \) where \( x_s \) and \( y_s \) are the UA position, and \( \psi_s \) is the heading angle. The dynamics are governed by \( \dot{\chi}_s = f_s(\chi_s, u) \) where \( u \) is the control input. This work considers the 2 dimensional non-holonomic kinematic model.

\[
\begin{bmatrix}
\dot{x}_s \\
y_s \\
\dot{\psi}_s
\end{bmatrix} =
\begin{bmatrix}
u_1 \cos(\psi_s) \\
u_1 \sin(\psi_s) \\
u_2
\end{bmatrix}
\]

\[
v_{min} \leq u_1 \leq v_{max} \quad |u_2| \leq \omega_{max}
\]

The control is done by computing the \( u \) for each UA that optimizes the objective function.

#### B. Sensor Model

The target being tracked is moving with the state vector \( x_t = [x_t, y_t, v_{xt}, v_{yt}, a_{xt}, a_{yt}]^T \) and an apriori estimate and covariance. The base station where the sensor fusion and estimation is done is stationary, with an exact position \( x_{b} = [x_b, y_b]^T \) known by both UA.

The sensors modelled in this network give a bearings-only measurement to the target from the UA at time \( k \) with the measurement function:

\[
z_k = h(x_t, x_{s,k}, v_{\theta,k}) = \arctan \left( \frac{y_{t,k} - y_{s,k}}{x_{t,k} - x_{s,k}} \right) + \nu_{\theta,k}
\]
where \( v_\theta \) is zero-mean Gaussian noise with covariance \( E[v_\theta v_\theta^T] = R_k = \sigma^2_\theta \). For multiple UA sensors, we simply add rows to this column for each sensor and augment \( R_k \) as follows:

\[
R_k = \begin{bmatrix}
\sigma^2_\theta & \cdots \\
\cdots & \cdots \\
\sigma^2_\theta_N
\end{bmatrix}
\]  

(23)

The \( H_k \) matrix described in the previous section is simply the Jacobian of the measurement function \( z_k \) and is defined as:

\[
H_k = \begin{bmatrix}
\frac{\rho_x - x}{\rho^3} & \frac{\rho_y - y}{\rho^3} & 0 & 0 & 0 & 0
\end{bmatrix}
\]  

(24)

### C. Estimation of Target Position

This work utilizes the Unscented Kalman Filter (UKF) [9] for sensor fusion and state estimation. The number of measurements is changing based on the network configuration and the number of packets reaching the base station. Changing the number of rows in the measurement function based on the number of measurements received will take care of the sensor fusion. If no measurements reach the base station, but propagation is required for the planner, the \( \sigma \)-points can be propagated to predict the next state and covariance.

For this problem, the vehicle will have unknown dynamics. To account for this in the filter, the Dynamic Model Compensation (DMC) technique [10] is employed. The position/velocity state of the vehicle is augmented with stochastic accelerations in the \((x,y)\) direction. The state vector estimated by the filter is then \( x_t = [x_t, y_t, v_{x,t}, v_{y,t}, a_{x,t}, a_{y,t}]^T \). The dynamic equation for the target is then written as:

\[
\dot{x}_t = \begin{bmatrix}
v_{x,t} \\
v_{y,t} \\
a_{x,t} \\
a_{y,t} \\
0 \\
0
\end{bmatrix}
\]  

(25)

The process noise will account for unmodelled dynamics such as acceleration/deceleration, or turns, through the stochastic accelerations in \( x \) and \( y \).

### V. Simulation and Results

#### A. Scenario

To test the algorithm described above, a scenario was set up with 2 UA with sensors sending information to a fixed base station where the data fusion and estimation occur. The UA are set to track a target with unknown dynamics. The target motion is seen in figure 3. The target makes two turns along with accelerating after the first turn, and decelerating after the second turn. The UKF with DMC described above deals with these unmodelled changes in state. The sensors on the two UA are bearings only with error modelled as white Gaussian noise with zero mean and \( 8^\circ \) variance. The sensors operate at 2Hz.

The scenario was run for 90s with 0.5s time step. The apriori estimate of the target state has an error of approximately 420m. The initial velocity is assumed to be zero.

The optimization is done in a receding horizon framework [11] with a planning horizon of 2s and a control fraction of \( \sigma = 0.25 \), resulting in a \( T_c \) of 0.5s. The probability of packet loss was modelled as a binary erasure channel where the probability is a function of range provided by a piecewise linear fit of empirical data from [12]. Figure 2 is a plot of the probability of successful transmission vs. range.

The algorithm was first run using perfect communication with no limits on range. This was to establish a baseline performance. Figure 3 shows the path of the target, both UA, and the estimate from the UKF. From the plot, the two UA are orbiting the target at approximately the minimum allowable distance. The two orbits are also offset from one another. To better illustrate this, three times were chosen (10s, 50s, and 85s). The optimal offset with perfect communication was shown to be \( 90^\circ \) [13]. The UA come close to maintaining that, however the unknown motion of the target along with solver accuracy skew this a little bit.

#### B. Perfect Communication

The next two simulations involve communication where some packets are lost. The first case was similar to the perfect communication case in that the planning did not take into account the limited communication, however the loss of packets was now modelled with an erasure channel. The case where communication was not taken into account was run to compare against the case where it is factored into the planning. When the base station received no measurements, the unscented transform was used to propagate the target state and covariance.

Figure 4 below shows the trajectory of both UA when communication is taken into account. Note the points of the
estimated state are plotted differently based on the number of measurement packets received at that step. Notice when the target is close to the base station, the UA orbit the target as seen with perfect communication. However, once the target gets to a distance where packets start to get lost, one of the UA starts to behave more like a relay (though still sensing the target). Though this is suboptimal based solely on sensor information gain, it does improve the expected information gain (described in section 3).

Fig. 3. Trajectory with Perfect Communication

Fig. 4. Trajectory with Limited Communication Taken into account

Figures 5 and 6 show the position residuals for both cases along with the 3σ covariance envelopes.

The total 2D position RMS’s are given for all three cases in the below:

<table>
<thead>
<tr>
<th></th>
<th>Perfect Communication</th>
<th>No Communication in Planning</th>
<th>Communication in Planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D Position RMS</td>
<td>3.77 m</td>
<td>4.50 m</td>
<td>3.95 m</td>
</tr>
</tbody>
</table>

The case with perfect communication performs best as expected. Also, of the two cases where packet loss is modelled, the algorithm which accounts for this in the planning performs quite a bit better. This is due to a difference in the number of measurements getting through between the two models. The table below compares the amount of measurement packets that got through between the two cases. Note that a total of 180 packets were sent by each UA over the entire run. The UA1 and UA2 rows show how many packets got through from each UA. The 'Both' row is the number of instances where the base station got both...
measurements, and the final 'None' row is when the base received no measurements, and had to propagate the state with the unscented transform. As expected the algorithm taking into account communication in the planning got more packets through to the base.

<table>
<thead>
<tr>
<th>Percentage of measurement packets received</th>
<th>Comm. in Planning</th>
<th>No comm. in Planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>UA1</td>
<td>85.0%</td>
<td>67.2%</td>
</tr>
<tr>
<td>UA2</td>
<td>85.6%</td>
<td>66.1%</td>
</tr>
<tr>
<td>Both</td>
<td>73.9%</td>
<td>52.2%</td>
</tr>
<tr>
<td>None</td>
<td>3.3%</td>
<td>18.9%</td>
</tr>
</tbody>
</table>

The next two figures, 7 and 8 show the plots of the actual probability of the erasure channel. Also note that an open circle signifies when a packet did not get through. As expected the second case keeps the probability quite a bit higher, which results in more packets getting through. The probability of success for UA2 stays high throughout due to multi-hop through UA1 back to the base. The plots also show the direct and multi-hop probabilities, where the final value is the maximum of these two.

Based on operational experience and FAA restrictions, it is not reasonable to fly more than this many aircraft anyways.

VI. CONCLUSION

It was shown for the active sensor network with two mobile sensors and one base station that incorporating communication through an erasure model into the path planning of the UA will decrease the estimation error by flying trajectories that reduce bit error rate while minimizing future uncertainty. It was also shown that for a relatively small number of sensors, network bandwidth will not be an issue.

There are several extensions to this work. The next step is to test this algorithm on the University of Colorado’s unmanned aerial systems. Another extension is to increase the number of sensors and targets. This can reasonably be done up to at least five sensing UA based on the bandwidth analysis.

Fig. 7. Probability of Packet Success. Communication is not considered.

Fig. 8. Probability of Packet Success. Communication is considered.

REFERENCES

[5] “Ieee 802.11g wireless lan medium access control (mac) and physical layer (phy) specifications-amendment 4: Further higher-speed physical layer extension in the 2.4 ghz band.” 2003.
A Collaborative Communication System Using a Mobile Robot in Wireless Sensor Network

Duck Young Kim, Doo Hyun Kim, Sung Ho Kim, Il Hong Suh, and Sung Ho Cho

Abstract—In this paper, we present a collaborative communication system using a mobile robot and operating in a wireless sensor network environment. In order to employ the mobile robot as a mobile sensor node in the WSN, a variety of issues, such as dynamic routing and dynamic topology management, is discussed. Besides a TinyOS module as a sensor node, the mobile robot is equipped with various sensors and various wireless communication modules for collaborative communications, so that the sensed data can be transmitted through a selected wireless module which best fits with the characteristics of data. The mobile robot also has an autonomous moving capability according to the order from the client. Several practical application scenarios are also presented using the mobile robot in WSN environments, and their usefulness and validity are demonstrated by the actual experimental setup.

Index Terms—Collaborative communication, WSN, Mobile robot, Dynamic topology management, Dynamic routing

I. INTRODUCTION

Wireless sensor network (WSN) consists of the small self-contained devices with computation, sensing and wireless communication capabilities. WSN plays a key role in the environmental hazard monitoring, the forest fire detection, the machine instrumentation, and so on.

In the past, the network only composed of the static nodes. However, mobile sensor network (MSN) that contains the mobile node is using now. It is expected to have higher utilization. Rapidly growing interests about MSN research is motivating the design of powerful test-beds to perform serious experimental investigations in such areas as the optimal node deployment, the network repair, the event detection, etc.

We consider the major features of MSN as three parts. One is a localization method for the mobile robot platform. The other one is current researches and utilities about mobile robot. The last one is the way to make a network with mobile node. Associate with first major features, the localization part: the main issues are accurate the localization of the sensor nodes and controlling mobility of sensor networks. To solve these problems, various methods are studied such as the RSSI based localization, dead-reckoning based localization and the computer vision based localization which analyzes the video images received by a camera [1]. The localization methods as well as the self-deployment which is one of the issues mentioned above are under researching too [2]. The mobile robot is able to have the information by interacting with the external network [3]. As a result of the rich resources, the mobile robot will be smarter.

Related with the second thing, mentioned above, an intelligent service robot is a different from general industrial robot. It has general robot functions as well as including integrated IT technologies, such as artificial intelligence, voice, visual, and emotion recognition. The robot can interact with human and provide various services such as housework supporting, education, and entertainment [4], [5]. Reference [6] is similar to ours but different from a number of mobile nodes and major topics which show the simulation data using their test-bed. In [6], WSN consists of two types of the motes which have different hardware specifications but has the same operating system, TinyOS. Each mote makes a sensor network respectively including two or more mobile nodes.

We designed a mobile robot which has a variety of communication modules. Since this module enables the mobile robot to connect external network, the mobile robot can have rich resources. Because of this, the mobile robot acts much smarter and has a lot of utilities. The mobile robot also supports autonomous driving by using the IPS. All components such as the communication modules, mobile robot platform and the sensors are controlled by main control system, is a personal computer in the mobile robot.

In this research, we focus on how to design the application software to adjust dynamic routing and topology. We also interested in how to carry out the scenario in the WSN using the mobile node. First, we describe the architecture of a mobile sensor network test-bed to perform dynamic topology management. After providing the architecture of mobile robot and the sensor environment details of our test-bed, we will explain about techniques for dynamically changed circumstance.

The remainder of this paper is organized as follows. In section II, we describe the mobile robot platform equipped with communication modules and sensor modules. The functions of each module will be explained. The following section III, we will show the method of dynamic routing and topology management for mobile node. In section IV, we shows that how to perform the several practical application scenarios and present the result of the application scenarios. Finally, we will offer some conclusions and describe our intentions for the future work in section V.

Authors are with Division of Electrical and Computer Engineering, Hanyang University, Seoul, 133-791, Korea (e-mail: dragon@hanyang.ac.kr).
II. A MOBILE ROBOT

A. Outline of the mobile robot

The Mobile robot platform which is designed is shown in Fig. 1.

The Mobile robot has various modules such as the WLAN module, the WiMAX module, the CDMA module, the Bluetooth module, the TinyOS module, the RFID reader and three Webcams. The mobile robot is composed of four floors. There are the webcams, tinyOS and IPS on the top of floors. The RFID, Bluetooth, WiMAX, WiFi and CDMA are located on the third floor. The personal computer for controlling communication modules is located on the second floor. On bottom of floor, regulators and batteries are located.

B. Configuration of the mobile robot

The system configuration of the mobile robot is shown in Fig. 2.

It consists of the following three parts: the mobile robot platform, the communication/sensor module part, and the application software part.

First of all, the mobile robot platform is basically responsible for autonomous and/or manually controlled movements. We presented the component model that is in the middleware. We developed the component of autonomous driving.

1) Robot server component

The robot server is a component to communicate with the mobile robot platform. It receives command packets from the application software and transmits it to the robot. Also, it transmits the state of the robot to the application software.

2) Navigator component

The navigator is the core component for autonomous driving. It recognizes the location of the robot and makes path plans. This component enables robot to move automatically.

3) Mapper component

The mapper is a sub component in charge of planning the path of the robot. Mapper generates a map about workspace. The map helps autonomous driving of the mobile robot.

4) Localizer component

Localizer uses coordinates received from IPS module to recognize the location of the mobile robot.

5) Pos2d component

The Pos2d is the class for movement in the 2 dimension space.

6) Pioneer component

This class is for the real action of the mobile robot.

7) Task Manager

This class is the component which manages tasks of the robot.

Secondly, the communication/sensor module part is composed of various wireless modules, such as CDMA, Bluetooth, TinyOS, WiFi, and WiMAX, as well as various sensors, such as webcams, IPS, and an RFID reader and...
antenna. Our test-bed encompassed kmote-s1. Each mote has an illumination sensor, a temperature sensor and a humidity sensor. Each mote carries an MSP430 processor running the Berkeley’s operating system, TinyOS. Fourteen motes except for the sink and the mobile node deployed in the hallway are located in test-bed. They transmit the sensed data to the sink node directly or through multi-hop. We can get an absolute position in our test-bed from IPS which is named as “StarGazer”. It uses infrared sensor to recognize the location. To use the IPS, Landmark, is a reflecting marks, must attach on the ceiling to reflect the infrared ray. The output of the position and the heading angle of the mobile robot are given with very precise resolution and high speed. Each Landmark has a unique ID and it can recognize the 2 meters in diameter. We used 131 different landmarks to cover the whole area and made a matching table to generate a full coordinate system in application software.

Finally, The CDMA module, Bluetooth module, and the IPS module exchange the packet with the server application software using RS-232C interface, WiFi, WiMAX and TinyOS modules can communicate with the server application software. The application software is in charge of the general managements of movements of the mobile robot and wireless connections of communication modules. As shown in the Fig. 2, the client part communicates with the server application software using the TCP/IP socket. The server application software also communicates with the middleware in the mobile platform using the TCP/IP socket.

C. Software layer

Application software is made up of two parts. The server application configuration is shown in Fig. 4.

![Fig. 4. Configuration of the server application software and the client application software](image)

The server application software mainly connects the client application software with the middle ware of the robot. Also, most modules, such as Bluetooth, IPS, TinyOS, and CDMA, in the robot platform are controlled directly by the server application software. The server application software has simple functions. It has the function setting up the server application’s IP address and connecting the socket to communicate with the client application.

The client application software is the main processor which handles most of tasks, such as calculating of the IPS coordinates for a robot accurate location, sending SMS to the user, detecting the sensor node which has unusual sensing data, acquiring the sensing data that the user needs, and so on. It can handle the RFID reader and webcams. The configuration of the client application software is shown in Fig. 4.

D. Floor plans of the test-bed

As shown in Fig.5, the development of the mobile robot was in progress on the 6th floor, IT building, Hanyang university, Seoul, Korea.

![Fig. 5. Floor plans of the test-bed. It displays sensor nodes position and hallway which the mobile robot can move.](image)

We designed the communication environment to cover the whole floor using the access point and three repeaters on the 6th floor.

III. A MOBILE ROBOT AS A SENSOR NODE IN WSN

The mobile robot having a sensor module is able to join the sensor network while it is moving. The robot plays a role as a mobile node in the sensor network to expand sensing area and it can perform some events such as a replacement of the node having a problem and fire detection depending on the circumstances.

Since WSN which we want to make includes the mobile node as well as fixed node, we required a routing algorithm that can support mobile node. Because moving range of the mobile robot is wide and there are a lot of walls which function as barrier, if the mobile node is recognized as a fixed node, its RF link will be disconnected because of its movement or the barrier. Therefore, we adapted minimum cost forwarding routing protocol provided by TinyOS. It allows the mobile node to join the network while it is moving.

This routing algorithm proposed by Fan Ye et al. [7][8]. This algorithm is considered for sensor networks since it does not require the storage of routing tables at sensor nodes. It established the optimal routing paths by exchanging the few message and it is scalable and simple to implement. The minimal cost paths file is established periodically. Once the minimal cost paths are established, all of the message traffic, which generated at the sensor nodes, are routed towards a sink node by forwarding along the minimum cost paths comprising one or more sensor nodes. By adjusting the time period that is
to establish the minimal cost paths, we updated the routing paths more frequently for the mobile node. Usually, in case the sensor network only makes up the fixed nodes, the renewal time of the minimum routing paths is 30 seconds or more. However, we set the renewal time to 10 seconds considering the moving speed of the mobile node. This setting enables the mobile node acting as the end node, source node, or the intermediate node to join the sensor network. Fig. 6 shows the dynamic topology. As the mobile robot is moving, the routing paths changes dynamically according to the link cost of each node.

IV. APPLICATION SCENARIOS

In this paper, in order to verify the features of the system based on the mobile robot, we experiment the scenarios to each situation. As a result, we verified that application software, the mobile platform, and communication/sensor modules worked well. There are figures which are captured from the demonstration video. There are six figures which described each scenario. In all of scenarios, the application software represents the topology in real time. As the mobile robot moves, it operates as the intermediate node or the source node. According to this, topology changes dynamically. In these figures, routing paths of network 1 was presented as a red line but routing paths of network 2 presented blue line in all scenarios figures.

A. Detection and replacement of a broken node

We will explain the software algorithm that is searching the broken node and replaces the node to the mobile node instead of broken node. Once the monitoring system starts, each node sends the sensed data to the client application every 2 seconds. The data that is sent contains the minimum routing path information of each node as well as the sensed data. Both of the data are inserted into the database. The method for checking an abnormal node is represented in Table 1. It uses the timer to check the absence time of the sensed data. According to the Table 1 techniques, the mobile robot receives the broken node ID from the client application software if the node cannot send its data in 30 seconds. Since the mobile robot already has the coordinates of all nodes, it can move to the broken node if the broken node ID is received.

![Table 1](image)

**TABLE I**

Algorithm1: Determine broken node

1. Node ID is a 1-15. (ID = 0; Sink, ID=100; mobile node)
   - input: the latest received data from DB
   - output: broken node ID
2. Initialize number of timers (is the same number of nodes)
   
   For i=1 to number of nodes {
   
   timer(i) = 0
   
   }
3. Database access to get the last data unique number(=code1)
4. Database access to get the current inserted data after the last data number (code1)
   
   Query = “select data, where code > code1”
   
   Execute the query
5. Print and check the data selected from DB (temperature, humidity, illumination)
6.1 Every node timer value incense per 1 second
   
   For i=1 to 15{
   
   Time(i) = Time(i) + 1
   
   }
6.2 If the node ID exists {
   
   Print data of temperature, humidity and illumination
   
   Timer(node ID) = 0
   
   }
6. Check the each node timer value whether it is over 30 or not
   
   For i=1 to 15{
   
   If (Timer(i) >30) Then
   
   Call send SMS function
   
   Call broken node display function
   
   Call send the broken node ID to mobile robot using TCP/IP
   
   Timer(i) = 0
   
   End if
7. Calculate distance between current position and broken node position
   
   cal_gap_x = Abs(Abs(broken_place_x)  –  Abs(current x coordinates))
   
   cal_gap_y = Abs(Abs(broken_node_y)  –  Abs(current x coordinates))
8. If mobile robot closes enough in broken node, it starts replacement
   
   Call change node ID function
   
   Call send the complete message to base-station using TCP/IP
   
   Call send SMS function

If the data of particular node don’t receive in 30 seconds, the node is identified as a broken node and it is replaced by mobile robot.

Fig. 7 shows that the mobile robot replaces the broken node. After replacing the node, the mobile robot covers the sensing area instead of the broken node.

![Fig. 7](image)
B. Participation in the other network

The mobile robot moves to the network 2 from the network 1 automatically. When the mobile robot arrives at the network 2, it sends arriving SMS to the user and changes its group ID (GID) so as to take part in network 2. The mobile robot transmits the sensed data to the sink node of network 2 as the source node. If the user sends a command to change a role of node, mobile node could be the sink or source node of network 2.

Fig. 8 represents scenario A, that mobile robot moves around the hallway in network 1. When it enters network 2, it joins the new network as a source node or a sink node.

C. Bridge mode between two networks

There are two different networks in our test-bed environment. These networks consist of nodes that have the same hardware specification but they cannot communicate with each other. The reason is that they have different GID. To connect these two different networks, we have to use a mobile robot. When the mobile robot is in the network 2, client application software orders to the robot that broadcasting the control message to nodes of network 2.

In Fig. 9, the mobile robot plays a role as a bridge node between two networks. Once two networks are connected, it is able to communicate with each other like one network by the mobile robot.

D. Fire notification and sensor information request

The server application software gets the sensing value from the database. During this process, if the server application software detects the temperature which is higher than the threshold value, the server application software can catch it. Then, the server application software sends the alert SMS to the user using the CDMA module. After that, the server application software transmits the coordinates of the place where the event occurs to the mobile robot. As soon as receiving the information of malfunction state, the robot moves to the point automatically. When it arrives there, the user can look around there through three webcams.

We consider the other scenario called remote environmental information acquisition of environment. The scenario is that the user requests the sensed data of the specific node. The user sends the SMS having the specific node number to the mobile robot, and then the server application software analyzes the SMS. After that, the server application software requests the database about information of the node. Once receiving the information successfully, the server application software replies SMS that has the specific information. Fig. 10 is the flowchart about these scenarios.

In Fig. 11, specific event occurs in the sensor network, the client application software detects the event. After recognizing the event, the client application software sends a command to the mobile robot in order to check the event. The mobile robot moves to the place where the event occurred, and then sends the video and SMS to the user.

Fig. 10. Flowchart of the scenario ‘fire notification and sensor information request’

Fig. 12(a) is the user’s mobile phone display when receiving the fire alarm notification SMS. Fig. 12(b) shows the picture verifying the sensor information request scenario. If the user want to know the temperature where node 6 is, the user send SMS, the content is ‘n06’, to the mobile robot, the mobile robot resend to user’s mobile phone SMS that contains the information of the node number 6.
V. CONCLUSION AND FUTURE WORK

We have developed and verified the test-bed using the various application scenarios. To prove these utilities, we made a sensor field in our laboratory building and a mobile robot platform which is equipped with communication and sensor modules.

Our future work is concentrate on making more intelligent network using the mobile robot. We have a plan to add two mobile robots. By using these robots, we can make a highly intelligent and cooperative network. If we use the mobile robot which is able to communicate with each other robot, the network efficiency will be improved in terms of expanding the network coverage, quick handling of abnormal events and the efficient environment monitoring. In addition, we are preparing the combining of heterogeneous networks using the mobile robot. In this research, we only use TinyOS system for WSN. The following topic, we will combine the TinyOs and ZigBee network using the mobile robots. We expect that the use of these mobile robots will enable the sensor network to improve the efficiency and the scalability.

ACKNOWLEDGMENT

This work was supported by Institute of Information Technology Advancement (IITA), under the R&D support program of Ministry of Knowledge Economy, Korea.

REFERENCES


DCOSS Workshop:

First International Workshop on Data Warehousing and Knowledge Discovery from Sensors and Streams
abstract—sensor networks are currently exploited to effectively monitor wide areas. To collect measurements describing the state of the monitored environment, different queries are frequently broadcasted to all sensors, providing the best possible approximation of the considered phenomenon. However, when querying a large number of sensors, the collection activity is characterized by high communication cost and energy consumption. Since sensors and sensor data are correlated both in time and space, a subset of nodes may be selected to model the network state.

This paper thoroughly describes the serene (selecting representatives in a sensor network) framework which provides high quality models for sensor networks. The models can be exploited to efficiently acquire sensor data by minimizing communication cost. serene exploits clustering techniques to discover spatial and temporal correlations which allow the identification of sets of correlated sensors and sensor data streams. Given clusters of correlated sensors, a subset of representative sensors, which best model each cluster, has been identified. Representative sensors are then queried instead of the whole network thus reducing communication cost and extending the sensor lifetime. Experiments performed on a real dataset demonstrate the adaptability and the effectiveness of the serene framework in providing energy-aware sensor network models.

i. introduction

Wireless sensor networks are currently used for a fast-growing number of different application fields. Among them habitat monitoring [1] and surveillance applications [2] are the most popular. While surveillance applications (e.g., health care monitoring [3], condition maintenance in industrial plants and process compliance in food and drug manufacturing [4]) alert the control system when a critical event occurs in an hostile environment, the habitat-monitoring applications (e.g., highway traffic monitoring, habitat monitoring like the monitoring on great duck island [5]) continuously monitor a given environment. In the latter case, to effectively provide a continuous monitoring, different queries are frequently broadcasted to all sensors thus collecting measurements describing the state of the monitored environment. However, when querying all sensor nodes, the data acquisition activity is characterized by high communication cost and energy consumption. Since sensor devices work under several constraints (e.g., limited power, high communication cost), new intelligent techniques for sensor network querying are needed. Main contributors to sensor energy cost are communication and data acquisition [6], hence this work has been focused on careful power management techniques for energy saving during data collection.

This paper thoroughly describes the serene (selecting representatives in a sensor network) framework which effectively identifies energy-aware models for sensor network data acquisition. serene exploits clustering techniques to study temporal correlation among sensor data streams and spatial correlation among sensors. Given clusters of correlated sensor data, the “best” subset of nodes, possibly including outliers, representing all sensors is singled out. Rather than directly querying all network nodes, only the representative sensors are queried to reduce the communication and power costs. Furthermore, since a query optimizer aims at identifying the cheapest execution plan according to an estimated cost, serene may be profitably exploited also in this context. Given a set of representative sensors identified by serene, a schedule minimizing the acquisition cost may be computed, for instance, by means of a tsp solver [7]. Experiments performed on a real public dataset show the effectiveness and efficiency of the serene framework in characterizing energy-aware models for sensor network data.

This paper describes the complete design and implementation of the serene framework, that analyzes historical sensor readings and allows the identification of energy-aware models for sensor network data acquisition. A preliminary description of a portion of the framework appears in [8]. The serene framework is also evaluated by means of an in-depth experimental validation that demonstrates the effectiveness of the proposed approach in discovering energy-aware sensor network models. Finally, this paper discusses the exploitation of different tsp solvers [9], [7], [10] to select the cheapest transmission schedule among representative sensors, thus minimizing acquisition cost.

The paper is organized as follows. section ii discusses related work, section iii presents an overview of the serene framework and describes the main features of its building blocks, whose implementation is discussed in section iv. In section v simulation experiments to validate the proposed framework are reported, while section vi draws conclusions and discusses future work.

This work is partially supported by the wifi4energy grant funded by Regione Piemonte.
II. RELATED WORK

Querying the sensor network entails the acquisition from sensors of measurements describing the state of the monitored environment. To provide the best possible approximation of the monitored phenomenon all sensors are queried \[11\], \[12\], \[13\]. However, this approach is characterized by high energy consumption. Since sensors are battery-powered and communication is the main source of power consumption, different approaches have been proposed to reduce communication costs.

The first technique is based on reducing the number of transmissions needed to answer the query \[6\], \[14\], \[15\]. To this aim, statistical models of real-world processes have been integrated into the query processing architecture of the sensor network \[6\], \[14\], \[15\], \[16\]. For example, the approach proposed in \[6\] inferred from all sensor measurements a statistical distribution for each considered phenomenon. The inferred model is then exploited to answer queries when the estimated accuracy is above a given threshold. Otherwise the query is redirected to (a subset of) the network, according to the required accuracy.

The second technique is based on reducing the number of queried sensors. \[17\] first proposed to select a subset of sensor nodes to represent the network (i.e., a snapshot of the network \[17\]). For the election process, nodes need to exchange a set of messages with their neighbors to elect the representatives of the surrounding environment. The selection process is driven by spatial correlation discovered among sensor neighbors, thus, only local similarities are considered. This approach has been enhanced by exploiting also temporal correlation among measures \[18\]. However, none of the approaches is able to detect correlation among faraway sensors.

The third technique is based on exploiting clustering algorithms to efficiently identify a subset of sensor nodes which best represent the network (e.g., the PREMON system \[19\], the LEACH system \[20\], and the CAG technique \[21\]). PREMON system \[19\] performs energy-efficient monitoring based on a clustered architecture. Cluster-head nodes compute the prediction model by exploiting MPEG compression algorithms. Successfully predicted sensor data are not transmitted, thus, energy consumption is reduced. The LEACH system (Low-Energy Adaptive Clustering Hierarchy) \[20\] assigns clusters based on the received signal strength and uses local cluster heads as routers. Furthermore, by means of a randomized rotation of local cluster-heads, the energy overhead among sensors is distributed through the network. The CAG technique (Clustered AGgregation algorithm) \[21\] discovers cluster of nodes by analyzing sensed measurements within a given spatial correlation threshold. This clustering remains valid as long as the sensor reading values are within a user-provided threshold (i.e., temporal correlation constraint). Only one sensor reading per cluster is transmitted, thus providing energy efficient acquisition and approximate aggregation of results within a user-provided error threshold. Any of the above approaches is efficient. However \[19\] requires the a priori knowledge of the cluster topology, \[20\] is not able to detect correlations among faraway sensors, and \[21\] transmits only one measure for each cluster. The SeReNe framework is more general with respect to the previous ones: (i) It does not require any a priori knowledge about cluster topology, (ii) it is able to detect correlations among faraway sensors, and (iii) it selects different R-Sensors from each cluster to better model the network state.

A parallel effort was devoted to the selection of the best plan to minimize the query execution cost \[6\], \[22\]. Correlation among attributes is exploited to identify the appropriate execution sequence. If two attributes are correlated, the execution plan always considers the attribute whose acquisition cost is lower. However, this approach queries the entire network.

One step further towards query optimization is based on exploiting query similarity to support efficient multiple query execution \[23\], \[24\]. Query similarity allows merging similar queries in a single query, which will be disseminated on the network. These approaches reduce the number of data requests submitted by the base station, and the total cost is lower bounded by the cost of a single query. When the query is disseminated through the network, this approach queries all sensors. Hence, efficient query execution strategies are still needed also in this case.

Finally, many studies in wireless sensor network research have been devoted to the design of efficient routing protocols. The aim of a routing protocol is to define the rules exploited to efficiently deliver data from the base station to the sensor nodes and from the sensor nodes to the base station. Two classes of routing protocols have been identified: data collection protocols \[25\] and dissemination protocols \[26\]. In the former, cluster based \[27\], \[28\] and tree based \[25\], \[29\], \[30\], \[31\] routing have been exploited, whereas in the latter, simple flooding techniques are typically used. While data collection protocols are used when all sensors are queried, dissemination protocols are used to broadcast the query to all network nodes. Another class of routing protocols is able to gather data from a selected set of nodes in the network \[32\]. These protocols are exploited when a subset of sensors is queried.

III. THE SERENE FRAMEWORK

SeReNe (Selecting Representatives in a sensor Network) is an environment for the identification of energy-aware models in sensor networks. Figure 1 shows the SeReNe framework integrated into a sensor network architecture. Sensor nodes frequently receive queries from the sink (i.e., base station). Each node performs the query over its sensor data and the result is sent to the base station by means of a multi-hop data collection protocol \[12\]. Before transmitting the query results, sensor data can be sometimes compressed by means of reduction/compression techniques \[33\], \[34\].

SeReNe generates sensor network models by means of two steps: (1) Correlation analysis, and (2) Selection of sensor representatives. The correlation analysis block discovers temporal correlations among sensor data streams, in terms of correlated sensors, correlation time and strength. Furthermore, it
Fig. 1. Architecture of the SeRENE framework

discovers spatial correlations among faraway and neighboring sensors. Given a set of correlated sensors, the second phase allows singling out a subset of sensors from the network, called Representative-Sensors (R-Sensors), which best represent all network nodes. During the selection of R-Sensors, several criteria may be considered (e.g., distance and transmission cost among sensors). The time window in which the sensor network model is effectively representing the network state is computed by guaranteeing a user-provided error bound \( \tau \) (see Section III-C for further discussion). Furthermore, the network model can be reliably adapted to network changes by continuously analyzing data collected through the network itself (see [8] for a further discussion).

Sensor network models generated by SeRENE are exploited to efficiently query the network. Smart sensors are characterized by computational, communication and sensing capabilities, thus allowing data processing inside the network. Each time a query is executed, only R-Sensors are queried, since they represent the network state for the corresponding query. To collect sensor data, the query and the execution plan are broadcasted to (interested) sensors. Hence, a transmission schedule needs to be generated. The transmission schedule generation block of the SeRENE framework (see Figure 1) aims at identifying an energy saving schedule which minimizes communication costs. Different optimization algorithms [9], [7], [10] have been integrated to identify an energy-saving execution plan (see Section III-D for further discussion).

### A. Correlation analysis

Correlation analysis on sensor measures detects relationships, both in the space and time dimensions, among physical phenomena and sensor data. SeRENE exploits clustering algorithms to perform correlation analysis among sensor data streams. These algorithms lead to a better identification of correlated groups even when sensors are physically far away and/or sensor data are collected in distant time instants (e.g., measurements collected in different days). Two different analyses are addressed:

- **Physical correlation analysis** allows discovering the similarity of the environment where the sensors are located. Two different cases can be discovered: (i) Two sensors located nearby sense similar values (e.g., sensors 1 in room A and 2 in room B, both at the second floor, sense the same value of temperature from 1 p.m. to 3 p.m. since they are in direct sunlight). (ii) Far sensors, located in similar environments, sense correlated measurements (e.g., sensors 3 in room C and 4 in room D sense the same value of temperature from 11:30 a.m. until 1:30 p.m. since both rooms are crowded for lunch).

- **Time correlation analysis.** Sensor data streams may be correlated over time. By means of this type of analysis, we can discover: (i) Correlated phenomena (i.e., two phenomena follow a similar pattern evolution) and (ii) correlated measurements of the same environmental parameter (e.g., the variation pattern of the measurement, for example, every hour). By means of the former relationship, we can query the sensor network for only one phenomenon, whereas the latter can be useful to decrease query frequency.

The effectiveness of clustering techniques in detecting sensor correlation is experimentally validated in Section V. Among the clustering algorithms (i.e., hierarchical, density-based, partitioning), density-based algorithms such as DBSCAN [35] have been exploited to perform the correlation analysis as discussed in [8]. In SeRENE, the distance between sensor data is measured by means of the Euclidean distance computed on normalized data.

By means of the DBSCAN algorithm, the SeRENE framework also addresses the outlier detection issue in sensor networks. Since sensor data are characterized by many outliers (i.e., either noisy measurements or independent sensor readings), outliers should be automatically detected and separately analyzed. The DBSCAN algorithm is able to automatically detect sensor data that do not fill into any proper cluster. These outliers are further analyzed to decide whether they represent correct measurements. Each outlier sensor that is eventually considered reliable may generate a cluster on its own.

### B. Selection of sensor representatives

This step consists in singling out a subset of representative sensors, denoted as R-Sensors, from a set of sensor clusters. The subset may contain one or more sensors for each cluster.
according to the required model accuracy (i.e., the error bound \( \tau \)). The number \( n \) of R-Sensors is set according to the required model accuracy. The number of representatives in each cluster is proportional to the number of cluster points. Reliable outliers are included in the R-Sensors.

Given a cluster of sensors, each of which senses \( k \) measures, the SeRENE framework exploits three selection strategies [8] to single out the subset of sensors that better model the correlated group.

- **Strategy 1**, also called *Measure trend*, is based on the analysis of correlated phenomena. Both the physical clustering in a given sampling time and the measurements collected during the considered time period are considered to represent physical and temporal correlations among sensors and their data. The best approximation of phenomenon \( j \) over the time period is given by the average value of measurements collected by all sensors during the considered period, denoted as \( \bar{M}_j \). Let \( \bar{M} = (\bar{M}_1, \ldots, \bar{M}_k) \), where \( k \) is the number of considered measures. Each sensor \( i \) is described by means of an array \( E(i) = (\bar{M}_{i,1}, \ldots, \bar{M}_{i,k}) \) computed over the time period. Representative sensors are the \( n \) nodes nearest to \( \bar{M} \) that correspond to the minimum communication cost.

- **Strategy 2** and **Strategy 3**, also called *Cluster shape* and *Cluster shape and core* respectively, are based on the physical location of correlated sensors. These selection techniques focus on cluster shapes of correlated sensors. Cluster border nodes are exploited to select representatives, thus detecting cluster shapes. At first, we compute the cluster barycenter \( \bar{x}_i, \bar{y}_i, \bar{z}_i \) = \( \left( \frac{1}{\#} \sum_{i=0}^{\#} x_i, \frac{1}{\#} \sum_{i=0}^{\#} y_i, \frac{1}{\#} \sum_{i=0}^{\#} z_i \right) \) where \( (x_i, y_i, z_i) \) are the spatial coordinates of sensor \( i \) in the considered cluster of \( m \) sensors. Next, sensor coordinates are normalized with respect to the barycenter by computing \( (x_i - \bar{x}, y_i - \bar{y}, z_i - \bar{z}) \) for each sensor \( s_i \). In this way, the barycenter is regarded as the reference system center. Distances between each normalized sensor and the barycenter are sorted in ascending order. A sensor is selected as representative if either its distance from all previously selected representatives is larger than its distance from the barycenter, or it is in a different quadrant with respect to all other representative sensors. Strategy 3 extends strategy 2 by including the closest-to-barycenter nodes as R-Sensors.

The effectiveness of the selection strategies in identifying a good subset of sensors to represent the network state has been experimentally validated and discussed in Section V-D.

### C. Sensor network model validation and characterization

Different clustering sessions are performed to suitably characterize sensor correlations. Each session considers a different set of measure combinations (e.g., temperature, humidity), and performs two analyses: (i) Correlation over time and (ii) Physical correlation. In both cases, clustering techniques are exploited. During the first phase, time series collected by each sensor are clustered. A set of correlated sensor values for each sensor are returned by the clustering algorithm. Clustering results are plotted on a mono-dimensional diagram where the x-axis represents time. On this graph, cyclically repeated time bands can be detected. Since the network consists of many sensors, clustering results obtained from each time series need to be merged, thus overlapping time bands are grouped together. For each group, the largest time band is considered to set the suitable validity window \( T_{model} \). A sensor network model has to be built for each time band, leading to different physical correlations for each band. A physical correlation clustering session is performed separately for each time unit in the time band, and analyzes the values observed by all sensors, resulting in a set of sensor clusters. Thus, correlated measurements collected in the same time period by different sensors are clustered into the same group, independently of the spatial position of the sensing device. Each cluster set is evaluated by computing the overall cluster validity [36] by means of a cohesion function, which is computed as the sum of the cohesion of individual clusters. The cluster set that maximizes the overall cohesion function will be exploited for building the sensor network model in the corresponding time band and for the selection of representative sensors.

After selecting the R-Sensors of a network state by means of one of the proposed strategies, the model validity window \( T_{model} \), which is the temporal interval in which R-Sensors provide a good approximation of the network state (i.e., the error bound of the result is \( \tau \)), needs to be computed. This is the largest subset of contiguous sampling times satisfying the threshold, given a subset of representative sensors. This time band, denoted as \( T_{model} \), is the model validity window and is computed as follows. At first, we estimate the approximate value \( \bar{M}_{T_j} \) of a measure \( j \) as the average on values collected by all representative sensors. The best approximation \( \bar{M}_{T_j} \) is the average on the values gathered by querying all sensors. For each subset of contiguous sampling times we count the percentage of contiguous samples in which \( | \bar{M}_{T_j} - \bar{M}_j | \leq \tau \). \( T_{model} \) is the largest subset of contiguous sampling times.

### D. Transmission Schedule Generation

The transmission schedule generation block of the SeRENE framework (see Figure 1) generates an appropriate schedule among R-Sensors to minimize communication costs and balance energy consumption among sensors. The transmission schedule is a list of sensor nodes to be visited to collect sensor data. It is the cheapest round-trip route that visits each node exactly once and then returns to the starting node (i.e., base station, the node that interfaces the query processor to the sensor network).

Since we focus on networks with known topologies and unreliable communication based on acknowledgment messages and retransmission, we can model the network by means of a graph composed by a set of nodes (i.e., sensors) and a set of edges [6]. Each edge between two nodes is characterized by a weight, which represents the average number of transmissions required to successfully complete the delivery. By considering \( p_{ij} \) and \( p_{ji} \) as the probabilities that a packet from sensor \( i \)
will reach sensor \( j \) and vice versa, and by assuming that these probabilities are independent, the expected number of transmission and acknowledgment messages required to guarantee a successfully transmission between \( i \) and \( j \) is

\[
\frac{1}{P_{ij} P_{ji}}.
\]

This value (i.e., the edge weight) can be exploited to estimate the transmission cost required to exchange data between \( i \) and \( j \).

Given the sensor graph, the transmission schedule that minimizes acquisition costs can be easily computed by means of a Traveling Salesman Problem (also called TSP) solver. Since solving the traveling salesman problem is NP-complete, many heuristics have been proposed in literature [7] that are known to perform well in practice. By means of one of a TSP solver algorithm we can select the schedule which minimize the communication cost and balance energy consumption among sensor. In particular, we integrated into SeReNE different algorithms to select an appropriate transmission schedule: (i) the Concorde algorithm [9], an exact TSP solver for symmetric TSPs based on a branch-and-cut approach; (ii) the Heuristic Concorde algorithm [7], an efficient and fast TSP solver using a branch-and-cut approach and the Chained Lin-Kernighan; (iii) the Genius algorithm [7], based on TSP heuristic and designed for the vehicle routing problem; (iv) the LKH solver [10], an efficient TSP solver using a branch-and-cut approach and Helsgaun’s-Lin-Kernighan-variant. These approaches have been experimentally evaluated in Section V-E.

IV. SeReNE implementation

The SeReNE prototype has been designed to be fast, stable, easy to access, and able to manage a huge amount of sensor data streams. Hence, the system has been developed with a modern, solid, and easily extensible architecture. Furthermore, an open-source approach has been selected. Technical choices include the Python programming language [37], the Django web application framework [38], the MySQL database management system [39], and Weka [40], a Java-based machine-learning toolkit. The SeReNE prototype results in a Weka-based web application which can be easily provided to any user through a web browser.

The SeReNE prototype allows the identification of energy-aware models for sensor network data by means of the following steps: 1) performing correlation analysis on both spatial and temporal dimensions by dynamically exploiting Weka clustering algorithms, 2) singling out the set of R-Sensors by means of the proposed strategies, which have been implemented in Python, 3) identifying the most efficient transmission schedule for R-Sensors by exploiting Phyton-developed TSP solvers. Finally, energy-aware sensor network models can be shown directly on the network map, providing the users with an intuitive visual feedback.

Furthermore, different techniques proposed in literature to (i) deal with missing data (e.g., non-parametric Expectation Maximization techniques [41], association rules mining to detect likely value replacing erroneous measurements [42]), and (ii) normalize data [43] could be easily integrated in the SeReNE framework to enhance its capabilities in handling sensor data streams.

V. Experimental results

We validated the SeReNE framework by performing different experimental sessions focused on analyzing (i) the effectiveness in detecting sensor correlation over time and physical correlation in space, (ii) the accuracy of R-Sensors in representing the network state, and (iii) the effectiveness in reducing energy consumption. Correlation analysis was performed by means of DBSCAN [35], a density-based clustering algorithm available in the machine-learning open-source environment WEKA [40]. Different sessions of analysis have been performed by varying the DBSCAN input parameters (i.e., a real number \( Epsilon \), and an integer number \( minPoints \), used to define a density threshold in the data space). Experimental results of the selection strategies highlight the effectiveness of querying representatives instead of the whole network. The error threshold provided by our model is often smaller than sensor accuracy. Experiments have been performed on an AMD Sempron(tm) 2400+ PC with 1666 MHz CPU and 512 Mb main memory, Linux operating system and WEKA version 3.5.2.

A. Experimental setting

We considered historical sensor data collected from 54 sensors deployed in the Intel Berkeley Research lab [44] between February 28th and April 5th, 2004. The dataset contains 2.3 million readings. Mica2Dot sensors collect temperature, humidity, light, and voltage values once every 31 seconds (epoch) by means of the TinyDB in-network query processing system [12], built on the TinyOS platform [45]. The \( x \) and \( y \) coordinates of sensors expressed in meters with respect to the laboratory map are also known.

The analysis of historical sensor data is preceded by a preprocessing phase, which aims at smoothing the effect of possibly unreliable collected measurements. Preprocessing entails the following steps: (i) outlier detection and removal, and (ii) standardization. Faulty sensors may provide unacceptable values for the considered measures. We removed data outside the validity range for each measure (e.g., humidity < 0 or humidity > 100) and the entire sensor data when at least two measures had been reported to be unacceptable. After this preprocessing step, the dataset contained 1.7 million sensor data. Finally, for each remaining measurement, values are normalized in the \([0,1]\) interval.

B. Correlation over time

The analysis of historical sensor readings over time has been performed by considering separately each time series collected by each sensor. We performed the analysis for every combination of the collected measures (e.g., temperature, humidity and light). Hence, different clustering sessions have been performed for each sensor data stream. For each session, the DBSCAN algorithm returns a set of clusters. Each cluster is a set of correlated sensor values.

By plotting the clustering results on a mono-dimensional diagram where the \( x \)-coordinate represents time, we identified
two/three cyclically repeated time bands, which always correspond either to daytime, or to night-time. The night band is shorter, and there is possibly an even shorter time band between day and night bands. Results shown in Figure 2 characterize the clusters obtained by considering three measures (i.e., temperature, humidity and light) sensed from two motes (mote 17 and mote 48). In both cases we identify two/three clusters. Since some sensor data have been removed during the preprocessing step, the number of samples is different for the two sensors. Overlapped time bands, identified by different clustering sessions, are grouped together. For each group, the largest time band is considered to define the validity window $T_{model}$. Hence, the largest time band for the nighttime and the largest time band for the daytime contribute to the corresponding ranges for $T_{model}$.

DBSCAN parameter estimation (Epsilon and minPoints) has been based on a large set of experiments. An Epsilon value suitable for all sensors has been selected (in the range 0.08-0.1), while the appropriate percentage of minPoints may be different for different sensors. When increasing Epsilon, the noise percentage decreases. However, in this case samples are grouped together in a big cluster. To provide better analysis capabilities, SERENe allows the user to choose a range of values for each parameter. Then, a grid parameter evaluation is performed, by varying each parameter at a time in the given range with a specific step. Finally, the user is presented with the results of each combination of parameter values, thus allowing an informed choice of parameter estimation. If the user does not specify any parameter range, then default values are proposed.

### C. Physical correlation

To study physical correlation, each clustering session analyzes all measurements collected from all sensors at a given epoch. Since some sensors could fail during the transmission or some values could be removed from the preprocessing step, some epochs report measurements for less than 54 sensors. We have run a large set of experiments on both the day and night time bands. For epochs belonging to the day time band, the following general trends have been highlighted: (i) During weekdays sensor data are grouped in a single cluster, which suggests that the lab is air-conditioned. (ii) During holidays, 3 or 4 clusters are formed, depending on the epoch. For epochs in the night time band, 2 or 3 clusters are created depending on the epoch.

Since the physical correlation analysis also addresses correlation among measures, we performed different clustering sessions considering different sets of measures. Figure 3 shows clustering results for sessions with different sets of measures. Temperature and light have been found to be highly correlated, as suggested by the presence of few clusters and a low percentage of noise. On the contrary, results considering all measures (temperature, light and humidity) lead to rather fragmented clusters and high noise presence, suggesting that humidity is less correlated with the other measures.

Figure 4 graphically shows the clusters of temperature and light measures obtained with Epsilon=0.1 and minPoints=6%. Figure 4(a) plots clusters on a Sunday around 6:30 p.m., while Figure 4(b) refers to a Tuesday night at 1:20 a.m. Inside the lab, three or four sub-areas with strongly correlated data can be identified.

### D. Accuracy of R-Sensors

To validate the effectiveness of exploiting R-Sensors instead of the whole network we analyzed: (i) Measurements collected by querying only R-Sensors with respect to querying the whole network, (ii) the relative error by querying representatives instead of the whole network, (iii) the mean square error of our model in $T_{model}$, and (iv) the mean square error when we exploited the model to query the network during the same temporal interval of $T_{model}$ in the following days.

Sensor data collected during the night of February 29$^{nd}$, 2004 in 12 hours of monitoring, has been used as the training set for the model. For the night time band sensor network model, representatives are selected from physically correlated sensor clusters related to the 8685 epoch. After the correlation analysis, the best epoch window $T_{model}$ ranges from 8524 to 9140 epochs (i.e., 5 hours and 28 minutes, from 11:57 p.m. to 5:25 a.m.). As default value, the number of representatives

<table>
<thead>
<tr>
<th>Mote Id</th>
<th>Sample #</th>
<th>Epsilon</th>
<th>minPoints (%)</th>
<th>Cluster #</th>
<th>Noise (%)</th>
<th>Elapsed time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>28132</td>
<td>0.1</td>
<td>1.25</td>
<td>2</td>
<td>10</td>
<td>2060.37</td>
</tr>
<tr>
<td>17</td>
<td>28132</td>
<td>0.08</td>
<td>1.25</td>
<td>3</td>
<td>14.3</td>
<td>2011.46</td>
</tr>
<tr>
<td>48</td>
<td>55079</td>
<td>0.1</td>
<td>10</td>
<td>2</td>
<td>21.16</td>
<td>5077.3</td>
</tr>
<tr>
<td>48</td>
<td>55079</td>
<td>0.08</td>
<td>10</td>
<td>2</td>
<td>34.79</td>
<td>5380.38</td>
</tr>
</tbody>
</table>

Fig. 2. Correlation over time - Cluster characterization

<table>
<thead>
<tr>
<th>Measures</th>
<th>Epsilon</th>
<th>minPoints (%)</th>
<th>Cluster #</th>
<th>Noise (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T L H</td>
<td>0.3</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>T L H</td>
<td>0.1</td>
<td>4</td>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>T L H</td>
<td>0.08</td>
<td>4</td>
<td>7</td>
<td>42.5</td>
</tr>
<tr>
<td>T L</td>
<td>0.3</td>
<td>4-14</td>
<td>1</td>
<td>2.12</td>
</tr>
<tr>
<td>T L</td>
<td>0.1</td>
<td>6</td>
<td>3</td>
<td>4.25</td>
</tr>
<tr>
<td>T L</td>
<td>0.08</td>
<td>4</td>
<td>3</td>
<td>12.76</td>
</tr>
<tr>
<td>T H</td>
<td>0.3</td>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>T H</td>
<td>0.1</td>
<td>4</td>
<td>4</td>
<td>6.38</td>
</tr>
<tr>
<td>T H</td>
<td>0.08</td>
<td>4</td>
<td>6</td>
<td>17.02</td>
</tr>
<tr>
<td>T</td>
<td>0.3</td>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>0.1</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>0.08</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 3. Physical correlation for Epoch=25157, sensor# 47 - Cluster characterization
has been set to 50% of the network sensors, while the error bound $\tau$ has been set to 0.4. The generated sensor network model provides information for queries on both temperature and light measures, either independently or jointly.

For the three strategies (described in Section III-B), Figure 5 shows the average temperature computed by querying only the representative sensors $T_r$ and by querying all sensors $T$ in different epochs in $T_{model}$. The three proposed strategies provide a good approximation of the monitored measures (see Figure 5) during $T_{model}$.

To validate the accuracy of the model, Figure 6 plots the relative error (i.e., $|T_r - T|$) introduced by querying representatives instead of the whole network. The accuracy for the temperature sensor$^1$ is also plotted to detect when our model is characterized by an error lower than sensor accuracy. The percentage of measurements affected by a relative error above the sensor accuracy is low for every proposed strategy, with the first strategy being the best (see Figure 6(a)).

Furthermore, to estimate the error introduced by the proposed network model, the mean square error is computed in $T_{model}$, given by $\frac{1}{\text{epoch}} \sum_{t \in T_{model}} (M_{rt} - M_t)^2$. $M_{rt}$ is the

$^1$Data available on user manuals accessible at http://www.xbow.com/Products/productsdetails.aspx?id=84.
average measure value computed by querying the representatives in a given epoch $t$ and $M_t$ is the value obtained by querying the whole network in the same epoch $t$. Figure 7 shows the mean square error for the three proposed strategies in $T_{model}$ (8524-9160 epochs) by varying the percentage of representative sensors. The first and third strategy provide a more accurate model than the second one, by leading to lower MSE values with few representatives.

To validate the effectiveness of the proposed model on different time periods, the network has been queried during the same temporal interval on the following days. Representatives have been exploited in each epoch included in the time frame corresponding to $T_{model}$. Figure 8 shows the mean square error for each strategy in different days (on each graph) and with different percentages of selected representatives (on the x-axis). The mean square error is comparable to the value obtained on training data. Hence, the R-Sensors provide a good approximation of the monitored phenomenon.

### E. Energy Consumption

In this section we analyzed the energy dissipated by querying only R-Sensors with respect to the whole network. The total energy consumed by sensors is computed by considering (i) the total number of transmissions required by all R-Sensors and (ii) the energy consumed to transmit data. The latter is obtained from the sensor data sheets (the radio link used on Mica2 motes and the Crossbow MTS400 [46] environmental sensor board).

Figure 9 reports the energy dissipated by querying R-Sensors compared with querying the whole network. The transmission schedule has been identified by TSP solver algorithms integrated in the SereNe framework. Experiments have been run by varying the ratio of selected R-Sensors. When the ratio of representative sensors ranges in 70%-95%, the energy consumption of R-Sensors is smaller than using all sensors. When a smaller subset of R-Sensors is selected, the communication cost significantly increases. This is mainly
Fig. 8. MSE on the temperature measure due to the decrease of the successful transmission probability. Since the number of retransmissions between two faraway nodes may be higher than that required among several close nodes, the corresponding energy consumption may increase.

Figure 9 also reports the energy consumption of each proposed strategy. For small percentages of R-Sensors (less than 75%), the measure trend strategy is able to select representatives whose usage saves energy during data collection with respect to the other strategies. For ratio values in the 70-95% range, all strategies lead to comparable energy consumption.

VI. CONCLUSIONS AND FUTURE WORK

The SeReNE framework provides optimized models for querying sensor networks while minimizing energy consumption for data collection. Given historical sensor data, clustering algorithms are exploited to identify groups of physically correlated sensors and groups of temporally correlated data streams. Hence, two correlation analysis dimensions have been discussed and exploited. Given a set of correlated clusters, a subset of representative sensors is singled out to optimally model the network state. Different TSP solver algorithms have been integrated into the SeReNE framework to select the best transmission schedule which minimizes the communication cost among sensors.

The SeReNE framework has been tested on a real public dataset containing measurements collected by sensors inside the Intel Berkeley Research lab. Experiments demonstrate the adaptability and the effectiveness of the proposed approach in providing energy-aware models for sensor network data acquisition.

Future developments of this work will address the integration of efficient turnover techniques for R-Sensors in the SeReNE framework. Conventional scheduling policies such as the Round Robin algorithm or LRU policies may be exploited to this aim.

ACKNOWLEDGMENT

The authors would like to thank the research team Wifi4Energy for technical contribution and Dante Bonino and Attilio Marchetti for developing parts of the SeReNE framework.

REFERENCES

Fig. 9. Energy consumption: transmission schedules performed by different TSP solver algorithms


Abstract—This research addresses the problem of energy conservation in WSNs. It proposes concepts and techniques to extract environmental information that are useful for controlling sensor operations, in order to enable sensor nodes to conserve their energy. These concepts and techniques are consolidated in a generic framework we term CASE: Context Awareness in Sensing Environments framework. CASE targets energy conservation at network level. A subset framework of CASE, we term CASE Compact, targets energy conservation at sensor node level. In this paper, we elaborate on these two frameworks, elucidate the requirements for them to operate together within a WSN and the WSN applications they can be applied to for energy conservation.

I. INTRODUCTION

The challenge of improving the sensors’ energy consumption is a key issue in wireless sensor networks (WSNs). This is because sensor network lifetime is directly related to operational lifetime. If sensors can operate for longer periods of time and the frequency of node failures can be reduced, the reliability and adaptability of the sensor network will improve. Furthermore, in keeping with Moore’s law [24], technological advances in sensor processing hardware significantly outpace progress in sensor battery capacity. Thus, it becomes imperative to enhance the energy conservation in sensor networks.

In this research, we focus on the challenge of maximising sensor network lifetime. In addressing this challenge, we note that one of the important factors that affect a sensor's energy consumption is the number of operations that sensors perform to gather data about their environment. For example, a sensor that frequently collects and transmits data will consume more energy than a sensor that only periodically collects and transmits data to a central processing station. We assert that the number of sensing operations can be reduced if certain sensors’ future readings can be inferred from previous readings. For instance, if it is known that all sensors in the same group give low temperature readings after a period of time (for e.g. deployed temperature sensors that allow us to assess fire situations in a building), we can choose to switch off some sensors in the group for the specified period to reduce the number of sensing operations performed. This assertion forms the basis of this research.

In this paper, we propose concepts and techniques to extract environmental information that are useful for conserving sensor energy. These concepts and techniques are consolidated in a generic framework we term the CASE (Context Awareness for Sensing Environments) framework. The CASE framework is designed for building, learning and triggering components for conserving energy in both centralised and in-network WSN configurations. CASE is further extended to create the subset framework, CASE Compact, to support autonomous and context-aware learning of triggers that control sensors efficiently in an ‘in-network’ data processing environment. As part of CASE Compact, we have proposed and developed a novel rule learning algorithm, Highly Correlated Rules for Energy Conservation (HiCoRE) algorithm, that can autonomously learn and discover rules to efficiently regulate sensing operations.

The remainder of this paper is organised as follows. Section 2 discusses existing data-driven approaches to conserve energy in WSNs, with a later emphasis on those that have studied context-awareness in WSNs. In sections 3 and 4, we discuss our proposed context-aware frameworks to target energy conservation in WSN. Following this, section 5 details how both frameworks can be used in a WSN. Applications that can be used with our frameworks are then described in section 6. Section 7 describes the experiments and the results obtained from our evaluation of CASE and CASE Compact. Lastly, we conclude our findings and suggest future work in section 8.

II. RELATED WORK

Existing data processing approaches have focused on the challenge of running sensor operations efficiently at multiple data levels. To conserve energy, these approaches have shown that achieving a suitable energy tradeoff between computation and communication operations is important. For instance, data-aggregation [17] reduces sensors’ communication by sending only summarised data rather than raw data. This tradeoff is achieved in these approaches generally at the network data level, node data level and the hybrid level.

At the network data level, the data model is such that sensor data arrives at the application running at a central
processing location, enabling the application to have a global view of data distribution within a sensor network. At this level, existing literature has addressed issues related to the energy efficiency of the sensor network system as a whole, targeting the efficient delivery of sensed data to the user (e.g., tinyDb [19]), efficient transfer of data between sensors (e.g., [22]) and efficient processing of collected data at the application (e.g., learning of spatio-temporal correlations [13]). To conserve energy, network-based approaches rely on information obtained from expert knowledge or machine learning and utilise the information to provide cues that would improve sensing and communication operations. For instance, in [21], the authors have utilised energy-maps that describe residual energies of sensors in the network to select the most energy-efficient route to send packets in the network.

Node-based approaches are approaches that focus on processing sensor data on a single sensor node that have higher processing capabilities. These are approaches that extract useful information from sensor stream data on the sensor node in a resource-efficient manner. For example, in [12], while utilising a hybrid system with mica2 and Stargate as higher resource devices, the authors were experimenting with integer-only Fast Fourier Transform (FFT) algorithm on micas in which the processing is done on Stargate. The benefits of using a computation hierarchy for in-network processing has also been supported in [18] where they justified performing in-network processing on an architecture comprising of macro-nodes in a bed of micro-nodes. In another study by [6], the authors propose a generalised Non-Parametric Expectation-Maximization (EM) algorithm tailored to operating on sensor nodes. In a more recent study, [20] investigated correlations that can be formed when sensors in loading trucks experience similar vibrations when the trucks send out the same load. The correlation information of the sensor nodes then allowed them to group trucks carrying out the same load. The unique contribution in their work lies in the incremental calculation of the correlation matrix.

In essence, these approaches have explored how sensor communication can be improved through utilising processed data obtained from sensors. However, these approaches have not yet explored how results obtained from computation can be used explicitly to drive energy-efficient sensor operations. In other words, the notion of using computed information (locally at sensor node or globally at application) to autonomously decide how a sensor should operate efficiently at any point during its sensing task has not been explored. For example, if the computed information suggests that sensing is no longer required at a sensed region, then the energy-efficient operation to be carried out by sensors in that sensed region is to sleep.

A data-driven technique that can provide direct control of sensor operations is necessary to further conserve energy in wireless sensor networks. From our observation of related work in data processing approaches, we discover that a subset of network-based approaches using context to make sensor operations more energy-efficient has research potential towards developing this technique. The context in their work refers to meaningful information discovered in a sensor network, in which context characterises sensed information by the sensors about their environments. In their research [7], [5], the authors derive contextual information about sensors to reduce the amount of sensing and communication required. For example, in [7], the authors focus on using spatio-temporal correlations in sensor data as contextual information about the network so that sensors can avoid sending information where possible, while in [5], work is presented on using low-power sensors to detect user intent in the application to save energy. Nevertheless, their work is not targeted towards energy conservation. For example, in [7], spatio-temporal correlations are used as context while context in a sensor network also covers other information such as remaining energy resource and sensed readings. Thus, this paper focuses on the problem of obtaining contextual information in a WSN to enable energy conservation. In the next section, we investigate our proposed context-aware approach in greater detail.

III. CASE FRAMEWORK

We propose the Context-Awareness for Sensing Environments (CASE) framework to achieve context-aware management and control for energy conservation in sensor networks. The CASE framework is built upon the notion of using available discovered contextual information in a sensor network to drive sensor operations efficiently, whereby contextual information can be regarded as any information that can be used to describe the situation of a sensor. The notion of context use in WSN was first presented in [3].

![CASE framework](image)

As illustrated in figure 1, CASE includes three main components: (1) Learning Component; (2) Context Processor; (3)

Sensor data readings that arrive at CASE are handled by the learning component and the context processor component. The learning component is responsible for discovering rules useful to define relationships between sensor readings to be used within CASE. These rules provide the basis for triggering to happen. As an example of a rule, given three sensor nodes $S_1$, $S_2$ and $S_3$ that detect temperature readings in a sensor network, a rule might suggest that when $S_1$ and $S_2$ have temperature readings over 30 degree celsius, sensor node $S_3$ would have a temperature above 30. The reason for the association between sensors $S_1$, $S_2$ and $S_3$ could be that they are all deployed in the same region where the situation context is summer and temperatures are high. A rule such as “$S_1 = 'H' \land S_2 = 'H' \rightarrow S_3 = 'H'” has two parts: (1) antecedents from the left hand side of the rule (i.e. $S_1$ and $S_2$ have high temperatures); and (2) consequents from the right hand side of the rule (i.e. $S_3$ has high temperature).

On one hand, the antecedents of the rule imply the context of the situation in which some action can be taken when CASE next determines that this context has occurred again. On the other hand, the consequents of the rule provide the action to be taken. In this example, when sensors $S_1$ and $S_2$ have high temperature readings, one action that CASE can take is to sleep “$S_3$” while the rule holds true. The rules would be stored in the Rules datastore to be used by the user or a script to program the Rules-Context datastore and Context-Action datastore in CASE. The rules can be obtained by mining sensor data through the learning component or they can be supplied by the user from prolonged observation of trends in sensor data:

(1) **Direct mapping from sensor readings using static rules by user.** The rules in the rules datastore can be manually programmed by users. The user can program the rules by observing the trends in sensor data stored over time. Manually programming rules will complement any other approaches in which the rules need to be updated frequently to adjust to any sensor data distribution changes. Nevertheless, incorrect rules can result in rule conflicts or context switches and as a result, possibly increase energy consumptions.

(2) **Learning Component mines for rules from sensor readings.** An adaptive approach to programming the rules can be provided through the use of a learning component to mine rules from sensor data. The learning component within CASE can be implemented in two ways to form rules: (1) by mining sensor data streams using existing stream association rule mining techniques such as [15] and [16] to track frequent items in sensor data streams and generate rules from the frequent items; and (2) by mining static sensor databases using conventional rule-mining techniques [1], [14]; this requires storage of sensor readings in databases.

The datastores used by the components within CASE stores information regarding contextual data, actions that can be executed and rules that describe conditions for which actions that are to be executed given the discovered context. These information can be more formally defined as events, conditions and triggers, commonly used for past active database systems [28]. Active database systems is a form of database technology that focusses on creating a system that would perform certain operations to reacts to expected events that can occur at runtime. The reactive nature of such a system is realised through the definitions of functions to be performed, storage for conditions to be met to trigger these functions and the expected events. As this research uses the same rule triggering paradigm as active database systems, the following discussion conceptualises the elements stored by the CASE datastores using some data definitions adopted by active database systems.

An active database system is centred on the notion of rule, and the rule can be defined in terms of three data parts: (1) **Event:** which could results in the triggering of a rule; (2) **Condition:** which is checked in the process of rule triggering; and (3) **Action:** which is performed when the rule is triggered and the conditions have been satisfied. These declared ECA rules (Event-Condition-Action rules) can then be applied in the process of events monitoring by an active database system.

In relation to our work, an ECA rule in CASE can be formally defined as follows:

**Event.** The event in CASE is the discovered contextual information represented by the context_tag $C_i$ in the set $C$, in which $C = \{C_0, C_1, \ldots, C_k\}$ is the set of context_tags that can be discovered in the system, where $C_0, C_1, \ldots, C_k$ are distinct and $|C| > 0$. $C_k$ is a string representation composed of numerical values (0-9) and alphabetic characters (a-z, A-Z). In other words, the context_tag is a user-defined textual representation used within the system to describe the current context for a given rule. This is given based on the user’s knowledge of the application and his interpretation of the learnt rules in the Rules datastore. The context_tag is used to complement rules that describe relationships among subsets of sensors and not all sensors in the sensor network.

**Condition.** The conditions in CASE are IF-ELSE statements that define the thresholds of sensor attribute values that need to be met to establish that any particular event has occurred. The operators used for the conditional statements include $<$, $>$, $\geq$, $\leq$ and $==$. The threshold values that need to be met apply to selected attributes belonging to the set of all possible sensor in the WSN, $S = \{S_0, S_1, \ldots, S_m\}$, where $S_0, S_1, \ldots, S_m$ are distinct and $|S| > 0$. $S_m$ in turn, can belong to a sensor group $G_o$ is in the set $G$, in which $G = \{G_0, G_1, \ldots, G_p\}$, $|G| > 0$ and $G \subset S$.

**Action.** The action in CASE is the action_macro $m_l$ in the set $M$, in which $M= \{m_0, m_1, \ldots, m_j\}$, $|M| > 0$, is the set of macros used that can be triggered in response to the event, or more specifically given any context_tag $C_i$. The macro $m_l$ can eventually be reduced to operations executed by sensors.

Information regarding the events, conditions, and actions can be stored textually or in any other metadata types such as XML. The relationships between context_tag, action macros
and operations is illustrated in figure 2.

Fig. 2. Relationship between context tags, actions and operations

As an example, let us reconsider the rule “$S_1 = 'H' \land S_2 = 'H' \rightarrow S_3 = 'H'". The following information is obtained from this rule: given the readings of sensors $S_1$ and $S_2$, there is a high possibility that $S_3$ would have a similar reading. This entails that some of the sensors deployed in the same region as $S_3$ can reduce their sensing frequency or be put to sleep because some abnormal behaviour of the sensor is unlikely, as long as the rule holds true.

Thus, from this rule, the context tag ‘SUMMER’ may be set to identify the event in which the condition ‘IF both $S_1$ and $S_2$ have high temperature values’ needs to be satisfied. If CASE detects this event and which satisfies the conditions, the assigned action macro, REDUCE_3, is triggered. Within the Context-Trigger Engine, REDUCE_3 is then decoded to the corresponding operation TRANSMIT_RATE&10000&3 to conserve energy. As a result, this particular sensor operation reduces the sampling rate to 10000 milliseconds for sensor $S_3$.

IV. CASE COMPACT FRAMEWORK

As an extension of the CASE framework, we propose the CASE Compact framework to target energy conservation at the node level. Using CASE Compact, the data is processed locally at sensor nodes that collect sensed readings through their sensing capabilities and from neighbouring sensor nodes. CASE is infeasible to be used directly on sensor nodes due to its use of heavy-weight learning and triggering components. Conversely, CASE Compact uses a lightweight rule-learning algorithm that we have first presented in [2]. It is designed to operate in both homogeneous and heterogeneous WSNs.

As illustrated in figure 3, CASE Compact is made up of two core components, namely the mining component and the triggering component. These two components are further described below:

1) Mining Component: The mining component is designed to parse and learn from sensor data arriving at central nodes. This is a node centralised model in which it is assumed that the central node has relatively higher resource capabilities than the other sensor nodes which send data to it. The notion that the central node has greater resource capabilities is necessary because by running CASE Compact, it will require additional resources to mine for patterns and do triggering. This data processing model is illustrated in figure 4.

For a homogeneous WSN, the node $M$ can be periodically nominated through a physical clustering protocol such as HEED [30] and LEACH [11] while for a heterogeneous WSN, node $M$ can be either a sensor node with greater residual energy or a PDA. The output of the mining component is rules, useful for triggering in a WSN application. These rules are generated via a novel rule-mining algorithm, termed HiCoRE (Highly Correlated Rules for Energy Conservation), that we have developed to work on sensor nodes. The novelty of HiCoRE is in its ability to extract only rules for frequent transactions that contain highly correlated sensor attributes; a high correlation between attributes is desirable because it signifies a strong relationship between attributed sensor nodes and in effect, increases the relevance of the rule generated.

We have demonstrated the effectiveness of using HiCoRE to conserve energy in [2] (referred to as ARTS).

2) Triggering Component: The triggering component is designed to allow central nodes to use the generated rules from the mining component to control sensor nodes in its group. Using the obtained rules with the triggering component, the central node can then send messages to power save sensors while the rules applied infer their values. This component is useful to conserve energy for the node sensing process in the applications that will be described in section V-B.
V. APPLICATIONS

A. CASE Applications

The CASE framework can be used to conserve energy generally for applications in which sensor data is processed centrally at the server. This refers to applications in which the base-station (e.g., laptop) collects sensor readings arriving directly from sensor nodes (circles in the figure) within one-hop communication or aggregated packets from sensor data sinks. One WSN application which we have explored is the use of the CASE framework to enable energy-efficient mobile data collection in WSN or data muling [4]. In general, data muling is an application that involves using mobile sensor nodes present in the environment to gather data from static nodes and deliver the data gathered to a central processing station. Data muling eliminates the need for intermediates to relay data and conserve energy.

The role of CASE to conserve energy in this application is by improving the mule detection process in data muling when upon discovery of useful contextual information in the WSN. Consider the arrangement in figure 5 showing a base station running CASE in the data muling application. In this figure, the data mules A, B and C serve to collect data from sensors 1, 2, 3 and 4 inside the shed and offload the gathered data to the base station.

Conventionally, a polling approach is used for the data mule to establish connection with the base station, whereby messages are frequently exchanged between the data mule and the base station. In contrast, CASE reduces the amount of transmissions necessary to establish this connection by using location information from RFID sensors to trigger a mule detection. In this scenario, with RFID tags attached to the data mules, once mule C enters the shed, the RFID readers located at the entrance of the shed will detect mule C on entry and send the detection information to the context-aware application. Through using CASE, the sensor nodes don’t need to “poll” for the mule but is told when they are near enough by CASE.

B. CASE Compact Applications

As shown in figure 6, we have identified three types of applications in which CASE Compact can be applied to. These include:

1. Triggering: for non-critical sensing applications or data-aggregation applications that can tolerate a coarse-granularity for sensed data, CASE Compact can be used to derive rules that control sensor operations efficiently using triggers [2].

2. Physical clustering: A physical clustering protocol for wireless sensor networks enables sensor nodes to be grouped based on physical parameters such as sensor nodes’ residual energy and node proximity. The use of a physical clustering protocol with a WSN allows the processing load to be balanced among sensor nodes, given at prior that all nodes carry equal processing capabilities and energy levels.

Through physical clustering, ‘cluster heads’ are selected to act as hubs for clusters of sensor nodes. The selection criteria used is dependent upon the physical clustering protocol applied. To ensure that suitable cluster heads are elected to represent their cluster groups at any time in sensing, data packets consisting clustering parameters are frequently broadcast by cluster members; also known as the reclustering process. The reclustering process is expensive due to the high cost of radio communication in a WSN.

CASE Compact can be used to reduce the energy costs to do physical clustering in the node reclustering process as well as the node sensing process:

2a. Node sensing process CASE Compact can be used to prolong sensor network lifetime by allowing cluster heads to run CASE Compact in parallel with the physical clustering algorithm to control cluster members, as illustrated in figure 7. The idea is that if cluster heads can infer readings values of members within their clusters, then the communication between cluster heads and their members can be reduced. For instance, in figure 7, if within cluster group G, a rule states that S1’s reading is always high when S2’s reading is low, then S2 can command S1 to reduce the frequency of its transmission to S2. Thus, CASE Compact would conserve overall sensor
network energies.

(2b) Node reclustering process The reclustering process for a physical clustering algorithm is expensive due to the frequent number of broadcast messages required to recluster. CASE Compact can be used to reduce the number of broadcast messages in this process by using the rule output derived from running HiCoRE to efficiently drive the iterative procedure in reclustering. In other words, the rule output would replace other external parameters used to determine the reclustering frequency, such as node residual energy used in HEED [31]. The rationale behind this is to enable reclustering to be performed in a manner favourable to controlling the node sensing process in CASE Compact as it prolongs overall WSN lifetime. In other words, the reclustering process should allow elected cluster heads to continually control cluster members if the rules generated at those cluster heads is favourable to conserve energy for their respective clusters.

In this regard, one rule output parameter that determines how much energy can be saved from rules generated by HiCoRE is the number of rule consequents in all the rules generated, i.e. the number of sensors that can be triggered. For example, the resultant probability for sensor nodes to recluster would thus be set as a function of the total cluster size for the cluster group,

$$\text{Reclustering prob} \leftarrow \frac{\text{clusterSize} \cdot \text{numberOfConsequents}}{\text{clusterSize}}$$

(3) Data Querying: A query processing system in a wireless sensor network is a system that allows users to interface a sensor network through queries in a high-level language (such as a variant of SQL) while abstracting the programming details required for users to query the sensors. Some current query processing techniques and systems for sensor networks include techniques that employ processing and routing strategies to optimise aggregate queries in sensor networks [26], approaches that use energy-efficient routing plans to answer queries [27], those that task the sensor network through declarative queries [29] and query processing systems such as TinyDB [19].

CASE Compact can be used to conserve energy for a data querying system through improving the:

(3a) Node sensing process Sensor nodes in a querying system such as TinyDB could run HiCoRE independently to conserve energy during sensing. One rationale for collecting sensing data locally is to enable queries that concern past sensor readings. In this case, the communication model would be such that sensor nodes are required to communicate information to selected central nodes (i.e. through data aggregation or physical clustering). This then enables learning and triggering to take place on the central nodes.

(3b) Node querying process The rules developed from using CASE Compact can also be used to improve the energy-efficiency to answer query requests. The idea is that if query requests are disseminated to central nodes that have learnt rules about sensors in their control, then rules can be used (to infer readings) in order to answer the requests without further posting the request to other child nodes. For instance, for our given example, consider that the following query is posted to node 1 to request light readings from all sensor nodes:

SELECT * FROM sensors, SAMPLE PERIOD 1s FOR 20 minutes

In this example, given that the central node has learnt of the rule ‘$S_1[H] \Rightarrow S_2[L]$’, the central node, when it continually receives a “High” light reading from $S_1$, can use the rule to infer a reading for $S_2$ (namely, “Low”), and so, need not request light readings from $S_2$, thereby saving energy in transmission. Similarly, energy would also be conserved when the rules obtained is used to answer approximate queries which involves MAX, MIN or AVERAGE aggregators.

Nevertheless, it needs to be noted that not all rules that are learnt can be applied to answer queries as not all sensors in a query can be involve in a rule. Therefore, rules to used in a query system would also need to be additionally filtered to the query being posted. As an example, if the query involves retrieving the value of sensor node $S_1$, then only rules involve $S_1$ as a consequent sensor can be applied.

VI. USING CASE AND CASE COMPACT IN A WSN

This section describes how both CASE and CASE Compact can be used together in a WSN for energy conservation. In particular, it details the network characteristics that would be present when both CASE and CASE Compact operate in the same sensing environment as well as the potential benefits and drawbacks in using both frameworks together in the same WSN application.

To begin, for CASE and CASE Compact to operate in the same sensing environment, the application in which the two frameworks are used together should embody the following characteristics:

Presence of a central server: a central server with a constant power input is required to run CASE in a WSN. This server is a high resource device such as a laptop or a PC. It should act as a central repository for sensed data so that contextual input from sensors can be processed and contextual patterns be learnt. The central server should also be able to send trigger messages to sensor nodes within its radio range.
Presence of central processing nodes: CASE Compact can be used in a sensor network either in a heterogeneous network with a static setup consisting of higher resource nodes and regular sensor nodes or a homogeneous network with a dynamic setup by using a mechanism to subgroup nodes in the sensor network and create temporary in-network group head nodes to control their respective subgroupings.

Utilisation of a communication hierarchy: a sensor network using both CASE and CASE Compact follows the communication hierarchy as illustrated in figure 8. The hierarchy depicts regular sensing nodes that are responsible for sending real sensory values to the intermediate nodes which runs CASE Compact and the results from these intermediate nodes are approximated values to be forwarded to the server that runs CASE.

Coarse-granularity sensing applications: coarse-granularity applications are targeted for WSNs running both CASE and CASE Compact. Some applications that we have discussed include event detection applications [9], [10], [25] and data aggregation [17]. Only coarse-granularity applications are applicable because sensor nodes running CASE Compact can only produce approximate values when triggering is used.

A WSN application that embodies the aforementioned characteristics can operate in the network arrangement that is shown in figure 9. In this arrangement, CASE Compact runs on selected central nodes (green dots), while CASE runs on the server connected serially to the base node M. The central nodes that run CASE Compact is responsible for energy conservation at the node level for local nodes under their control, while the server the runs CASE is responsible for energy conservation at the group level for the sensor groups formed. For instance, in figure 9, sensor nodes S0, S1 and S2 belong to the single sensor group G1. CASE controls the sensor group G1, while S2 running CASE Compact controls nodes S0 and S1. The triggers coming from CASE can apply to any of the sensor groupings that have been formed in this network.

To enable power savings in this setup, two requirements are imposed on the central node. Firstly, the central node needs to run CASE Compact and in doing so, execute the HiCoRE algorithm to process real values coming from the nodes in its group and perform triggering to power save controlled nodes. The approximated values collected by any central nodes are then forwarded to the server, enroute node M. Secondly, a central node needs to respond to any trigger messages coming from the CASE server. Specifically, this response depends on the type of trigger commands the central node has received from the node M:

1. **Sleep trigger:** if the trigger message is to sleep the central node and any other nodes that it is controlling, the central node would first need to sleep the nodes under its control before sleeping itself on a timer finally. When the central node has awakened, it would first broadcast wakeup messages to the sensor nodes that it has slept previously and reinitiate transmissions from those nodes. The transmission to node M from the central node then resumes to its default operation.

2. **Reduce activity trigger:** the trigger message to reduce the activity level of a sensor group entails either lowering the transmission frequency or reducing the load of data transmissions. In both cases, reducing the activity level would entail the central node sending messages to the nodes in its group to inform the nodes of the intended transmission frequency or load (i.e. the message packet structures to send next) while the central node itself would send data less frequently or less data to node M.

3. **Default sensing trigger:** the process to revert sensor groups to default operation follows the same process that described to reduce the sensing activity. This resumes the central node and its members to the default state programmed.

It can be anticipated that using both CASE and CASE Compact in this WSN arrangement allows more energy to be conserved, relative to only using either frameworks individually. This is because utilising both CASE and CASE Compact enables energy conservation at both node levels and application levels. At the sensor node level, patterns are learnt locally at central node to coordinate sensor operations in local groups and at the application level, more general network behaviours can be captured to drive operations for larger...
groups of sensors.

VII. EVALUATION

In this section, we evaluate the performance of CASE and CASE Compact implementations where the results obtained are presented and analysed. The aim of the experiments conducted with CASE is to evaluate energy conserved through using discovered contextual information at the network level to control sensor node operations. This objective has been achieved through using our CASE implementation in a pig sty scenario. It needs to be noted that though these results have appeared in [3], [2], this is the only work that has given a complete explanation of the integration of the two frameworks, expressed our results in greater detail and in that context.

A. CASE evaluation

The aim of the experiments conducted with CASE is to measure the energy saved when CASE has been utilised to drive sensor node operations in a pig sty scenario. It needs to be noted that the purpose of the experiments performed is to demonstrate that energy can be conserved (through applying triggers such as sampling rate adjustment) when a WSN system adapts to various types of context, and not simply to show the effects of varying sampling rates on a WSN system. Thus, we have not presented any formal procedures for determining the exact sampling rate required but rather, this has been left to the application designer.

In this scenario, we envisage a pig sty with deployed mica2 sensor nodes to collect sensory information about the living environment of pigs. This sensory information includes light, temperature and voltage readings. As illustrated in figure 10, the setup involves deploying a static node $S_1$ within the pig sty, a mobile node $S_4$ attached to pig A and a base node $S_0$ attached to the base station. It has been assumed that both $S_1$ and $S_4$ are within communication range of $S_0$. The objective is to use contextual information discovered to enable the energy-efficient control of sensor $S_1$ installed inside the pig sty. The contextual information in this case is primarily derived from data readings communicated to the base station by sensor nodes $S_0$, $S_1$ and $S_4$. More specifically, the contextual information pertains to the movement/location of the pigs.

![Fig. 10. Pig sty scenario](image)

In these experiments, we consider the following context_tags used within CASE:

1) Normal. This represents the default context of the sensors.
2) Pigs_Out. This represents the context whereby pig A with sensor $S_4$ attached is outside the pig sty.
3) Pigs_In. This represents the context whereby pig A is inside the pig sty.
4) Daytime. This represents the context that it is daytime and pig A is outside the pig sty for an extended period of time.
5) Nighttime. This represents the context that pig A will be inside for an extended period of time.

![Fig. 11. Pig sty experimental results](image)

Results from the experiments performed using various types of triggers are recorded from actual runs (given in figure 11(d)) as well as simulations using PowerTOSSIM [23] for each experiment individually. The PowerTOSSIM results are shown in figure 11. The purpose of the actual runs is to collect results pertaining to the total number of packets sent and received at the base station, whereas the simulations serve to measure the energy consumption of the sensor node that has been contextually controlled by CASE (in this case, sensor $S_1$). The data collected from running PowerTOSSIM are filtered using Perl scripts and presented as a graph using gnuplot. These experiments and the analysis of the results obtained are further discussed as follows:

1) Experiment 1: control experiment: This experiment serves to measure the energy consumption of a sensor node when CASE is not utilised to drive sensing operations. This experiment is used as a reference comparison in terms of energy savings for subsequent experiments. In this control experiment, sensor nodes $S_0$ (mote 0), $S_1$ (mote 1) and $S_4$ (mote 4) are placed in room temperature and under moderate lighting conditions. As CASE is not running, no context changes are detected to occur over a duration of the 20 minutes...
run. Motes 0, 1 and 4 collect light, temperature and battery voltage readings at every 1000ms and transmit the collected data packets to the base station. The default context_tag in CASE is ‘Normal’. In total, 608 data packets were sent out to the base station by all sensor nodes. Figure 11(a) shows the cumulative energy consumption of sensor node $S_0$ over time. Sensor nodes $S_1$ and $S_4$ exhibit the same behaviour with respect to energy consumption. As a whole, the graph result demonstrates that the energy consumption of a sensor node is directly proportional to the duration of runtime when no other context_tags have been detected in this experiment.

2) Experiment 2: sampling rates experiment: In experiment 2, CASE is used to control sensor node $S_1$ by dynamically adjusting its sampling rate upon detection of a suitable context_tag. In this experiment, we first assume that $S_1$ can sample at three different rates within CASE, i.e. at 1000ms, at 10000ms and at 100000ms. Let us also assume that the current context_tag in CASE is ‘Pigs_In’ and $S_1$ is sampling at the rate of 1000ms. When pig A with $S_3$ attached moves out of the sty, high light readings detected by $S_4$ outside enables the detection of the context_tag ‘Pigs_Out’ by CASE. As a consequence, CASE sends the trigger to reduce the sensing rate of $S_1$ in the sty to 10000ms. Subsequently, the sensing rate of $S_1$ is returned to 1000ms when the pigs return to sty and the context_tag is ‘Pigs_In’. The results shown in figure 11(b) illustrate the energy consumed by $S_1$ during the course of triggering by CASE to react to the current context_tag detected. In both figures, this correspond to the context_tag ‘Normal’ during the time period from between 0 and 360 seconds, the context_tag ‘Pigs_Out’ between 360 and 640 seconds, and context_tag ‘Pigs_In’ between 640 and 1200 seconds.

A total of 406 data messages were sent out from all sensor nodes in this experiment. In empirical terms, the results obtained show a reduction in the total number of messages sent in the network of 33.2% ((608-406)/608)*100 as compared to the control experiment, with 12 packets written out to node $S_1$ from the base station to change its sampling rate. It also demonstrates an energy saving of around 22.9% (77800-60000)/77800 when we compare the final energy consumed at the time = 1200 seconds in figures 11(a) (i.e. 77800) and 11(b) (i.e. 60000). Despite the energy conserved, it can be noted that some additional energy is expended when $S_1$ needs to readjust its operations to the trigger resulting from a new context_tag, also termed context switch. These context switches occur at times 340 and 580 seconds, incurring time lags of 20 and 60 seconds respectively. From this observation, it is likely that, if context switches occur too frequently, a significant amount of energy may be expended. A possible solution is to set appropriate thresholds within CASE to limit the number of context switches, which is a possible direction for future work.

3) Experiment 3: message size experiment: This experiment serves to measure possible energy savings from using triggers in CASE to reduce the size of data messages transmitted by sensor nodes. In this experiment, we assume that two possible message structures may be set, i.e. a 24-byte data packet that encapsulates light and temperature data (packet type A) versus a 16-byte data packet that excludes the light data (packet type B). Let us also assume that the current context_tag is ‘Daytime’ and all light readings currently need to be recorded. However, when the lights inside the sty are switched off (at night) resulting in the context_tag ‘Nighttime’, light readings are then no longer required at $S_1$. This results in the trigger by CASE to temporarily disable light readings, i.e. the preference to choose the packet type B over the packet type A.

Figure 11(d) shows the results from our experimental run of 7 minutes when this situation happens. As message size changes are not observed in PowerTOSSIM simulations, corresponding graphs are not presented. Nevertheless, the results from the actual runs may be observed. In this run, the context_tag detected for the first 2 minutes is ‘Daytime’, switching to ‘Nighttime’ in the next 3 minutes and back to ‘Daytime’ until the end of the experiment. During ‘Nighttime’, $S_1$ transmits data packets without light readings at the rate of 1000ms to the base station. In total, 207 packets were received from sensor nodes at the base station during this period. Out of the 207 packets recorded, 41 packets had the packet type in listing 6.2 of 16 bytes and 166 have the packet type in listing 6.1 of 24 bytes. This equates to a total of 4640 bytes received at the base station using this trigger. In comparison, if all 207 packets sent were of the packet type in listing 6.1 of 24 bytes, 4968 bytes would have been sent from the sensor nodes, implying an additional 328 (4968 - 4640) bytes which is not required, based on the current contextual information. This achieves a 6.6% reduction in data transmitted. This is a conservative estimate as the payload only omits the light readings. It is anticipated that more data transmissions can be reduced if more elements of the readings are omitted, for instance, both light and temperature readings. Furthermore, this is with respect to one sensor. A typical WSN consists of a large number of sensor nodes, for instance, up to one thousand nodes in Grape Networks [8]. Finally, it must also be considered that that the time duration of the experimental runs is lesser than a real-world deployment where sensors would continuously monitor for several days. The implications for such scenarios clearly indicate that the CASE approach can have significant benefits.

4) Experiment 4: sleep node experiment: This experiment serves to measure energy savings from applying triggers to sleep sensor nodes by CASE. This involves putting sensor nodes to sleep over an extended periods of time to conserve energy when it is expected that sensing is not required. The adapted scenario is similar to that of the message size experiment in which the context_tags ‘Daytime’ and ‘Nighttime’ are used. Contrary to that experiment, however, $S_1$ is put to put sleep instead to further conserve energy until readings from $S_4$ indicate that it is ‘Daytime’ again. Figure 11(c) shows the energy consumption of node $S_1$ during the context switches, i.e. ‘Daytime’ between 0 and 240 seconds, ‘Nighttime’ between 250 and 630 second and ‘Daytime’ again from 640 seconds onwards. Figure 11(c) shows the cumulative
energy consumed in this period. In comparison to the control experiment, approximately 67.9% (77800-25000)/77800 is conserved when we compare the final energy consumed in figures 11(a) (i.e. 77800) and 11(c) (i.e. 25000). Although this experiment demonstrates that significant amounts of energy is saved when the trigger is to sleep a node, the resulting impact of the trigger to sensing operations within the WSN needs to be considered. Firstly, a slept sensor node would be completely unresponsive for some defined time period. This is not feasible when the sensing task is mission-critical. Secondly, a sensor node that has been put to sleep needs to be awakened either through an internal timer or by a subsequent trigger due to another context tag. In the latter case, the onus is on CASE to ensure that the sensor node put on sleep will be awakened.

B. CASE Compact evaluation

The aim of the experiments conducted in this section is to measure the amount of energy saved when CASE Compact is used with a query processing application. The experimental setup used is modelled upon the scenario discussed in section V-B that describes how CASE Compact conserves energy when used with the query processing system. TinyDb. The experiments performed here measure the energy that can be conserved in a scenario involving the querying of a node. The experimental setup is illustrated in figure 12.

![Experimental setup](image)

As shown in figure 12, this setup involves the base node $M$ connected to a laptop and other sensor nodes $S_0$, $S_1$ and $S_2$ that transmit light, temperature and microphone readings to node $M$. In this setup, let us assume that the user creates the query at the base station. In the process of answering the user query, node $M$ requests for sensory readings from sensor nodes $S_0$, $S_1$ and $S_2$. Upon hearing data requests from node $M$, the sensor nodes then transmit their sensory readings to $M$, en route to the base station. The transmissions from sensor nodes to $M$ continue for the lifetime of the query.

For the purpose of this evaluation, let us now consider that node $M$ runs the HiCoRE algorithm and has obtained rules relating to $S_0$, $S_1$ and $S_2$ prior to running a query. The obtained rules can then be used by $M$ to only request sensory readings from sensor nodes when the readings cannot be inferred by filtered rules, for instance, rules with high confidence values. In theory, this should reduce the amount of data communications necessary to answer a query and thus, conserve energy. In order to demonstrate that energy can be conserved, the experiments that follow study how much data transmissions can be reduced in this manner and the compromise on the data quality obtained, if any, through measuring the accuracy of the rules obtained. In order to measure rule accuracy and the actual number of bytes transmitted, computer simulations are performed. This involves running a Perl script to determine the amount of data transmitted in this scenario for the lifetime of a hypothetical query when our C implementation HiCoRE is used/not used. For the run with HiCoRE, synthetic sensor data transactions are first piped to the HiCoRE program so that rules may be learnt at prior. The synthetic data set used has the following properties:

1) $S_0$ light readings and $S_1$ light readings have a positive correlation of $0.8 \pm 0.04$.
2) $S_1$ light readings and $S_1$ temperature readings have a positive correlation of $0.8 \pm 0.04$.
3) $S_2$ light readings and $S_2$ temperature readings have a negative correlation of $-0.8 \pm 0.04$.

This dataset is run on HiCoRE for a period of 8 minutes. It is noteworthy that this evaluation is performed on a synthetic dataset as this is the way we can establish the accuracy of HiCoRE by testing whether the rules predicted capture the known correlations. After 8 minutes, any rules generated by HiCoRE are stored in $M$’s memory. Before node $M$ requests sensory readings from sensors $S_0$, $S_1$ and $S_2$, it will first look at all the rules that have been stored in memory. If a rule in memory meets the confidence threshold, then the rule will be used. Specifically, $M$ will use the inferred consequent values of the rule rather than requesting the necessary values from the sensors.

For both simulation runs, the query used is “SELECT * FROM sensors, SAMPLE PERIOD 1s FOR 7 minutes”. The results obtained from running this query in the simulations are presented and analysed in the following sections i.e. on the accuracy of rules generated and on the resulting data throughput for query with/without rule adaptation.

1) Measuring accuracy of rules: Table I shows the rules obtained when HiCoRE is run on the aforementioned synthetic dataset for 8 minutes with a set confidence threshold of 0.5. It also shows the resulting confidence of the rule generated. In order to measure the accuracy of the rules generated, each of the rule is then applied to perform prediction on the same dataset used for the query evaluation. In this regard, a prediction is considered to be successful when the predicted value is the same as the expected value in the dataset in discrete form. As an example, given the rule R1, '$S_1 L[H] \rightarrow S_0 L[H]\$', if the light value of $S_0$ is correctly predicted to be low when $S_1$ has a high value, then the prediction is successful. For rule R1, the prediction is successful 34 times out of 41 times when $S_1$ has a high light reading, giving a success rate of approximately 82.9%.

Two observations can be made from the accuracy results shown:

1) The rules generated by HiCoRE are able to capture the correlations between data attributes, if they are present in the dataset. This is demonstrated through the results
showing that a majority of rules generated reflect the relationship between the light readings of $S_0$ and $S_1$ as well as between the light readings of $S_1$ and temperature readings of $S_1$.

2) Rules with high confidence values generated by HiCoRE typically imply a high success rate in prediction. This is further illustrated in figure 13, whereby rules such as R5 having a confidence value of 0.5 (normalised to 50% in figure 13) has a resulting success rate of 45.3%. Thus, confidence is a good measure of the predictive accuracy of the rule for data querying application.

2) Measuring data transmitted: Figure 14 illustrates the amount of data transmitted in bytes by sensors to $M$ when the query is answered in three different situations:

1) No rules are used. In this situation, $M$ requests sensors $S_0$, $S_1$ and $S_2$ to send their readings to $M$ every second for 7 minutes.

2) Filtered rules are used. In this situation, $M$ selects high confidence rules to be used in answering the query. For the purpose of this evaluation, let us assume that rules R6 and R7 are used by $M$ due to the fact that they have high confidence values i.e. 1.0 and that they are not in conflict with one another i.e. the consequent of R6 rule is not the antecedent of rule R7 and vice-versa.

3) Randomly selected rules. In this situation, $M$ just randomly selects two rules to be used from all the rules stored in memory. Let us assume that, in the worst case scenario, low confidence rules R5 and R8 have been selected.

From the results obtained in figure 14, it can be observed that energy is conserved when rules generated by HiCoRE have been used to answer the query, regardless of whether random rules or filtered rules have been used. This is evidenced by a reduction in the overall amount of data communicated which directly impacts on the overall energy consumption by the sensor network [23]. Specifically, the adaptation to filtered rules reduces data transmission by 22.7% (1450-1120)/1450, while the adaptation to random rules reduces data transmission by 41.4% (1450-850)/1450. While it may appear that using random rules is better than using rules filtered/selected using confidence levels, it needs to be highlighted that accuracy is reduced with random rules. In particular, it needs to be noted that the resulting error rates from prediction using random rules are much higher than using filtered rules. Empirically, when filtered rules R6 and R7 are used, the combined success rate is 91.7% (i.e. referring to table I, total combined errors of 4 divided by total combined results predicted of 48 from using rules R6 and R7) while when random rules R5 and R8 are used, the combined success rate is only 50.6% (i.e. referring to table I, total combined errors of 43 divided by total combined results predicted of 87 (53+34) from using rules R5 and R8).

In general, the results obtained demonstrate that the data transmissions required to answer a query is reduced (and in effect, energy is conserved) when CASE Compact is used in data querying. The best strategy to select rules for use within a query application is not necessarily selecting rules that conserve the most energy, but rather, using a strategy that can achieve the best compromise among criteria such as the overall energy conserved, error rate that can be tolerated by the application (for instance, if the application is mission-critical, then the success rate/confidence threshold should be high) and appropriateness to the current query (for instance, using rules relating to sensors that are relevant to the query).
VIII. CONCLUSION AND FUTURE WORK

In this paper, we have presented our CASE framework to conserve energy in a centralised data processing environment. The experimental results have been evidenced the potential for energy savings that can be achieved through CASE and CASE Compact. However, it needs to be noted that the current experiments presented have not tested the scalability of our approach, i.e. real world WSN applications typically run up to thousand of nodes and real world implementations present other issues such as radio link failures and node failures. By itself, the experimental results serve to and have only demonstrated the benefits of embodying context-awareness in sensor network and node levels through the sampling and data querying applications. In future work, we hope to further validate the concept of context-awareness through larger scale testbed implementations and experimentation with more WSN applications.

REFERENCES

Fully Decentralized Recommendations in Pervasive Systems: Models and Experimental Analysis

L. Becchetti, U. Colesanti, A. Marchetti-Spaccamela, and A. Vitaletti
“Sapienza” Università di Roma
Dipartimento di Informatica e Sistemistica “A. Ruberti”
via Ariosto 25, 00198 - Roma, Italia,
Email: {becchett,colesanti,alberto,vitale}@dis.uniroma1.it

Abstract—In this paper we propose and investigate the effectiveness of recommendation techniques based on collaborative filtering for use in fully decentralized, pervasive systems of small devices with limited communication and computational capabilities. We consider a reference scenario where items are distributed in a shop (e.g. a bookstore or a supermarket) and are tagged with such a device (e.g. RFID). We do not assume any cooperation among users and, more importantly, we assume that each device stores only aggregate information over the histories of users that visited the item in the past. Information exchange among nodes is thus mediated by users collective and unpredictable navigation patterns. Our algorithms do not require any explicit interaction among users and can be easily and efficiently implemented. We analyze the theoretical behavior of our recommendation strategies and assess their performance in practice, both by simulation using synthetic data and on real, publicly available datasets.

I. INTRODUCTION

Distributed, local algorithms are the natural solution to many challenges of pervasive computing. Nowadays, the pervasive deployment of tiny devices with minimum storage and limited or no computational capabilities appears a realistic perspective; one major obstacle are the strict energy constraints of battery powered devices. We refer to a new class of passive devices (i.e. not powered by batteries), the most prominent example being RFID tags, that has emerged in the last decade. An RFID tag is an object that can be applied to or incorporated into a product, animal, or person for the purpose of identification and tracking using radio waves. Passive RFID tags have no internal power supply. The tag draws power from the radio waves emitted by the reader, namely no batteries are required and consequently the energy issues can be neglected paving the way to new pervasive solutions. However, passive tags have minimal or no computational power, the memory is limited to only few KB and the distance between the reader and the tag is at most 10 meters (ISO 18000-6), but it is typically only few centimeters (ISO 14443).

In our envisaged scenario, RFID reader devices carried by customers suggest recommendations only on the basis of the succinct information obtained interacting with the tagged objects during the visit to a shop. In this context the stream of data (from tags to readers and vice-versa) is determined by users’ visit patterns and information exchange among nodes is mediated by the collective and unpredictable navigation of users. Our main goal is to show that even under the stringent constraints outlined above, it is possible to design effective fully decentralized solutions.

In the last ten years recommendation systems have been recognized as an important research area and much work has been done both in industry and academia on developing new approaches. As a result, a number of recommender applications are used in a variety of e-commerce systems, e.g., for recommending books and other products by Amazon [15], [1], movies by MovieLens [17], DVDs by Netflix [2]. One important feature that is exploited in many proposals is the ability of storing all available information in a centralized system and/or exploiting active collaboration with the user or among users in order to classify them, so that users belonging to the same class have similar taste. We also remark that some of the proposed solutions are computationally expensive; therefore heuristics have been proposed to deal with large data sets.

The future mass deployment of pervasive sensor networks opens the possibility of new scenarios for recommendation systems. For example, we refer to MyGROCER [13], a recent proposal for a ubiquitous computing environment for supermarkets based on a smart shopping cart that exploits shopper’s identity to provide a personalized service. As observed in [20] (see also [3]), improvements are necessary to extend recommendation systems to new scenarios, “including [...] products to purchase in a store made by a smart shopping cart”.

We believe that mass deployment and privacy issues deems for new models and algorithms able to extract useful information without requiring user’s identification and cooperation. This is not the only constraint: the limited computational capabilities of sensors and the high cost of communication should also be considered. All the above constraints pose the challenge on whether sensors are able to extract the distributed knowledge of the network in order to provide meaningful information with no central communication and at limited costs.

Distributed non-cooperative recommendations

Our goal is to assess the capabilities of a network to self
provide good recommendations to shoppers without relying on a centralized supervised system and/or on active cooperation among shoppers. We assume that items of interest are distributed in a shop (e.g., a bookstore or a supermarket) and are tagged with a passive RFID, namely a device with limited storage, communication and computational capabilities.

Each user is characterized by a (unknown) ranking of items, describing her preferences: when a user enters the shop she receives a smart shopping cart/basket (i.e., an RFID reader) that stores her history during the visit and has some computation capability. This is a quite reasonable assumption as RFID readers can at present be integrated into PDA or NFC smart phones. Assume that a user approaches/selects item \( i \) at some point. We assume that \( i \) stores in the RFID tag a suitable summary of the histories of users that visited \( i \) in the past. When item \( i \) is visited by user \( j \) then the smart reader recommends a new item (or a set of items) of potential interest to \( j \), using current summary at \( i \) and \( j \)'s history. The recommendation of an item is good if the proposed item scores high in \( j \)'s preferences.

Users visiting the shop transparently exchange information with smart tags. In particular, a user visiting an item reads its current statistics and uses this information and her history (i.e., the list of items visited so far) to construct aggregate statistics to recommend items. Figure 1 depicts a possible scenario: every item has an associated smart tag with a unique integer ID. In the picture, the agent is visiting item 6 after visiting items 2 and 4. The reader uses the statistics stored in node 6 and the user’s history to provide a recommendation (e.g., item 1 has score 3, while item 3 has score 21 and is thus the top item in the summary).

Note that i) items of interest are tagged with computational devices of limited computational capabilities, ii) we do not assume any cooperation among users iii) users are not identified (as is done in Amazon) and, more importantly, iv) we assume that there is no explicit communications among devices tagging the items and therefore each has a partial view of the past history of the system, determined by users’ past visit patterns. We remark that these limitations comply with privacy issues and that the scenario we consider is technologically realistic; it is closely related to architectures proposed for smart shopping carts [13] and smart shelves [8].

Following a common assumption in the literature [3], [12], [14], we consider a model in which items are partitioned into disjoint clusters, each representing a different topic; we assume that each user \( j \) is characterized by an unknown vector, describing \( j \)'s preference for the different clusters, whereas choices within a cluster depend on the popularity of items. In this way, we naturally model many realistic scenarios. For example, consider a bookshop: different users can have varying degrees of interest towards different topics (e.g., science fiction vs cooking) but within a topic, there are very popular items (i.e., best-sellers) and less considered ones.

Results of the paper
We are interested in studying how, despite the above limitations, the system is capable of providing good recommendations. We show how simple, computationally inexpensive heuristics allow to effectively profile users and provide good recommendations in the scenarios described above.

The general recommendation algorithm we consider is simple: when a user visits item \( i \) she is recommended the item scoring highest in \( i \)'s summary among those not yet visited. The core of the whole problem is defining scores that can be efficiently maintained locally at every node and that effectively reflect users’ unknown preferences. In general, scores and rankings can statistically depend on users’ visit patterns in complex ways. We tackle this issue by defining models of user behavior that statistically describe users’ preferences and visit patterns in terms of unknown preference vectors over the item set.

Experimental analysis also shows that - not surprisingly - the number of examples required for achieving good recommendations strongly depends on the size of the item set and the number of users: in particular, we need to accumulate an amount of information at each item (i.e., visits by users) that is polynomial in the size of the set of potentially interesting items. This means that statistics collected at popular items will sooner provide an accurate view of the system than those collected at less popular ones. A further contribution of the paper is to provide asymptotically tight lower and upper bounds on the number of examples required by the recommendation algorithms to compute good estimations of item scores and thus provide good recommendations. These results show that the number of examples necessary to provide accurate scores for all items might not be feasible in all practical cases (this might for example be the case when the item set is very large or it is dynamically changing over time). One natural possibility to overcome this limitation is to recommend not just a single item but a small set of items; in this case a recommendation is successful if the user likes at least one of the proposed items.

Accordingly, the third contribution of this paper is to analyze algorithms that recommend a small set of items instead

Fig. 1. Example of user scenario
of just one. In order to assess the effectiveness of the proposed algorithms, we use both simulations and raw data provided by Netflix, a popular on-line DVD rental service. Results show that even the recommendation of a small set of items (say three or four) has a good chance of containing at least one item that will eventually be visited by the user. Surprisingly, results on real data show that the quality of recommendations provided by our algorithms is much better than suggested by simulations. This aspect is discussed in detail in Section IV.

This paper represents a step in the tantalizing topic of providing good recommendations in distributed settings. We believe that much more work is needed. Our study emphasizes some issues that deserve further study; we discuss them in the final section.

Related work.
There is a vast literature in the area of recommendation systems and we refer to the excellent survey [3] and to the references therein for an overview of the main applications that have been considered and for a thorough presentation of the different methodological approaches.

In [10], Deshpande et al. compute similarities between items and obtain ratings to determine the top-N item recommendations. Namely, an \( m \times n \) user-item matrix \( R \) stores binary information on users’ choices: \( R(i,j) = 1 \) if the j-th customer has purchased item \( i \) and zero otherwise. The top-N recommendation problem is defined as follows: given a user-item matrix and a set of items \( U \) that has been purchased by a user, identify an ordered set of at most \( N \) items \( X \) such that \( X \cap U = \emptyset \). Using matrix \( R \) items are classified and user is suggested a set of items similar to items in \( U \). We remark that their approach is centralized, while in our case the user-item matrix is not available when a suggestion is given.

The formal analysis of recommendation systems has received attention over the past few years. Singular Value Decomposition (SVD) [16] was shown to be useful to cluster items; we remark that SVD is computationally intensive and requires centralized information and often requires additional conditions for its applicability that are not met in practical cases [11]. In [6], [11] the goal is to approximately recover the latent structure of users’ preferences. However, proposed solutions require extensive data on each user and a centralized and computationally expensive computation.

Kumar et al. [12], [14] study the off-line problem where preferences are identified with past choices; items are clustered and each user has a probability distribution over clusters: a user first chooses a cluster by her distribution and then chooses a product uniformly at random from that cluster. The goal is to recommend an item from the user’s preferred cluster.

In [5], a distributed solution is proposed to on-line recommendation in which a user is in search of an item she likes. The algorithm is randomized: at each step, the user either selects an item uniformly at random or asks another user about her preferences. It is shown that if most users can be classified in \( k \) types (where users in the same type share preferred items) \( O(m+n) \) step are sufficient to find preferred items. Although this solution is distributed, active cooperation between users is required. A similar remark applies to [4] where the goal of the users is to learn their complete preference vector (approximately) while minimizing the cost of probing.

II. MODELS OF USER BEHAVIOR

In our scenario, every user \( j \) enters the shop, visits\(^1\) a subset of the items and then leaves the system. The identity of users is not stored, hence multiple visits of the same agent to the shop are not tracked and are modelled as independent visits performed by different users. We assume that an agent visits each item at most once during its permanence in the system. The alternative model in which a user may perform multiple visits to the same item is also interesting (e.g., this may be a typical pattern in an amusement park) and will be considered in further research.

We assume that every item has a unique integer identifier \( i \in [n], \) where \( [n] = \{1, \ldots, n\} \), \( m \) denotes the overall number of users. In the sequel, we use the terms user and agent interchangeably. The same will hold for the terms item and node.

Our scenario entails two aspects: i) the choice of the items to visit and ii) the order in which the items are visited.

Cluster based item selection. We assume that every user \( j = 1, \ldots, m \) has an associated vector \( w_j = (w_{1j}, \ldots, w_{nj}) \), \( w_{ij} \), called user profile in the sequel, describing \( j \)'s potential interest for item \( i \). Note that \( w_j \) is unknown to the system. In particular, \( w_{ij} \leq 1 \) gives the absolute probability that user \( j \) will select item \( i \). Note that profiles completely define users’ interests and the selection of one item does not affect the choice of another one. Such a model assumes no dependence among user profiles and is too general and unrealistic in most cases of interest for applications, in which users appear in classes characterized by similar tastes and preferences.

Following common assumption in the literature [3], [12], [14], we assume items are partitioned into disjoint clusters, \( C_1, C_2, \ldots, C_s \). We further assume that each user \( j \) is characterized by a preference vector \( (p_{1j}, \ldots, p_{sj}) \), where \( p_{kj} \) denotes the preference of user \( j \) for items in cluster \( C_k \). We also assume that, within a cluster, the weight of each item is the same for all users. Namely, we model user preferences by assuming that for item \( i \) and user \( j \), \( w_{ij} \) satisfies the following:

\[
 w_{ij} = p_{kj} w_i, \quad i \in C_k
 \]

We call this the cluster model. Accordingly, we call \( p_{kj} \) the weight of cluster \( k \) for user \( j \) and we call \( w_i \) the cluster weight of item \( i \). Note that, since \( p_{kj} \) is the absolute probability that user \( j \) visits cluster \( C_k \), \( \sum_{k=1}^{s} p_{kj} \neq 1 \) in general.

We also assume that each item is aware of the cluster it belongs to. Though a restriction, this assumption models scenarios of practical interest, such as a bookshop, an e-shop

\(^1\)The meaning of “visiting an item” is crucial and depends on the application. Here, by visit we mean an active and detectable interaction between the agent and the item. E.g., browsing through the pages of a book, purchasing an item in a supermarket etc.
or a supermarket, where items are (physically or virtually) arranged in groups defined by some notion of similarity (e.g., topic or use).

**Order of visit.** We consider two possibilities. In the *random visit model*, we assume that, provided both item $i$ and item $h$ are visited by agent $j$, they have an equal chance to precede each other in $j$’s visit. More formally, denote by $S$ the subset of items selected by agent $j$. Then, in the random order model, for every $i, h \in S$, the probability that $i$ precedes $h$ in $j$’s visit is $1/2$.

In the *weighted visit model*, we assume that, conditioned to the event that agent $j$ visits a subset $S$ of the items and provided $j$ has already visited a subset $X \subset S$ of them, the probability that the next item visited by $j$ is $i \not\in X$ (where, of course, $i \in S$) is proportional to $w_{ij}$. This implies that, if $l = |X|$, the probability that $i$ is visited in the $l+1$-th step of $j$’s visit is $w_{ij} / (1 - \sum_{r \in X} w_{ri})$. We again point out that this is the probability conditioned to the event that $j$ visits subset $S$ of the item set.

The weighted visit model describes a strong dependence between user preferences and visit. To consider the bookshop example, many people are likely to be first attracted by popular, recently published books in their fields of interest. It is clear that none of the models we consider here will fully describe a real scenario. The models we consider are the ends of a broad spectrum of possible alternatives. The right model depends on the application and is likely to borrow from both.

The above model is intended to strike a balance between simplicity and soundness. It is clear that this choice brings to some simplification with respect to the scenarios of potential interest: the recommendation based on items’ popularity can be sensitive to changes of users’ visit patterns over time; furthermore assuming that rankings inside clusters are uniform with respect to users (i.e., they only depend on item popularities) may be unrealistic in some scenarios. Finally, visit patterns might depend on different (e.g., geometric and physical) constraints. Nevertheless experimental analysis on real Netflix dataset, shows that at least in the scenario the data refer to, even this simple model captures important trends. For example, we observe in section IV that a significant correlation exists between the order of users’ visits and items’ popularities within the same movie cluster.

### III. Recommendation Algorithms

As we have already observed, recommendation occurs at every node upon a user’s visit. This is done by computing at each item $r$ a ranking of all items belonging to the same cluster as $r$, in order to recommend the most interesting ones. Ideally, we want every node $r$ to locally maintain a suitable monotonic function $f(\cdot)$ of cluster weights such that $f(w_r) \geq f(w_h)$ if and only if $w_r \geq w_h$. Furthermore, $f(\cdot)$ should be estimated in a completely self-organized fashion, from quantities that are measured locally at $r$ or are carried in the histories of users visiting $r$. To achieve this goal we assume each user carries a vector $\mathbf{H}(j)$, whose $i$-th component $\mathbf{H}_i(j)$ contains the piece of $j$’s history that pertains item $i$, if $j$ visited $i$, is null otherwise. It is clear that, since each user only visits a small subset of a potentially large set, $\mathbf{H}(j)$ should be stored in compact form in a practical implementation, which we actually do. We only consider the definition given above to the purpose of simplifying notation.

We also assume that each item (i.e., node of the system) $r$ keeps a summary of histories of users that visited $r$ in the past. The amount and kind of information is crucial for the practical feasibility of each proposed solution. In particular, nodes tagging items are likely to be inexpensive devices with limited storage and processing power in any realistic applications. To address this issue, we restrict to solutions in which a generic item $r$ can store a vector (or an equivalent representation thereof) $\mathbf{R}(r)$, whose $i$-th entry $\mathbf{R}_i(r)$ is some numeric value used to rank item $i$ in $r$’s picture of the system’s past history, or null whenever $r$ possesses no information about $i$.

A crucial aspect is the probabilistic nature of the available information: since the set of items and the order of visit of a user is a random variable, $\mathbf{H}(j)$ is the outcome of a random process and this implies that $\mathbf{R}(r)$ is generated from statistics over the histories of users that visited $r$. As we shall see, this implies that the information available at item $r$ does not allow to exactly compute function $f(\cdot)$ as described previously in this section; in fact, one of our contributions is algorithms to compute $\mathbf{R}_i(r)$ from statistics collected at $r$, so that $\mathbf{R}_i(r)$ provides a good estimate of $f(\cdot)$.

**Recommendation.** The general recommendation algorithm is the obvious one and is modular with respect to how ranking of items is computed: upon $j$’s visit, $r$ recommends to $j$ the $\hat{T}$ top ranking items in $\mathbf{R}(r)$ that have not yet been visited by $j$. The overall operation of a node/item is summarized in Figure 2, with **UPDATE**( $r$, $j$) implementing the core operation of item ranking.

**Require:** Parameter $\hat{T}$: number of recommendations
1. When agent $j$ having history $\mathbf{H}(j)$ visits node $r$ then
2. $\mathbf{R}(r) = \text{UPDATE}(r, j)$
3. Recommend $\hat{T}$ top elements of $\mathbf{R}(r)$ not present in $\mathbf{H}(j)$

Fig. 2. Recommendation algorithm.

We next discuss how $\mathbf{R}(r)$ is computer at each node $r$. Computation of $\mathbf{R}(r)$ clearly depends on the model we assume to describe user behavior. In the rest of this section we only consider the weighted visit model, since this is the model proving more effective in capturing real user behavior on the

---

\(^2\) It is clear that even storing $O(n)$ bits at every node can be overly expensive and streaming techniques [19] can prove useful to reduce storage use. This is an implementation issue and will be the focus of future work, our primary goal in this paper being to assess the feasibility of distributed recommendation algorithms under the models we consider.
Conceptually, there is no difference.

The key point is that

The result of Theorem 1 also describes the convergence properties of our algorithms. It is possible to prove that these bounds are asymptotically tight. A complete analysis is not the purpose of this paper. For the sake of completeness, we briefly address the simpler aspect of the estimation of $E[F_r(t)]$ at a generic node $r$. It is possible to prove the following theorem, whose rigorous mathematical statement and proof are given in the appendix:

**Theorem 2:** Assume the cluster based model. For every $i \in C_k$ and for every $0 < \delta < 1$, $\Theta\left(\frac{\ln n}{m(t)}\right)$ are necessary and sufficient to estimate $E[F_r(t)]$ accurately.

Note that accurately estimating $E[F_r(t)]$ is crucial, since it appears in the explicit expression of $f(w_i)$.

Fig. 3. Update algorithm.

\begin{algorithm}
\caption{Update algorithm.}
\begin{algorithmic}
\State \textbf{Require:} node $r$, agent $j$
\State 1: $r$ maintains a vector $w(r)$ of estimates of nodes' weights.
\State 2: $F_r = F_r + 1$ \{A new agent is visiting $r$\}
\State 3: $H_r(j) = F_r \{j$ must initialize $H_r(j)\}$
\For {i = 1 \ldots n} \{and $i \neq r$\}
\If {$H_r(j) > 0$} \{j visited $i$\}
\State $N_{ir} = N_{ir} + 1$
\State $R_i(r) = \left(1 + \frac{H_r(j)}{F_r}\right) N_{ir}$
\EndIf
\EndFor
\State return $R(r)$
\end{algorithmic}
\end{algorithm}

real datasets we consider. Very similar result hold for the (simpler) random visit model. For the sake of space, these will be given in the full version of the paper. Next, we give some additional notation that will be used in the sequel. In particular, we denote by $n$ the number of items, by $m(t)$ the number of agents that entered the system up to time $t$. We denote by $F_i(t)$ the number of agents that visited item $i$ up to time $t$. We set $V_i(r, j) = 1$ if $j$ visits $i$ and $r$ in this order, 0 otherwise. Finally, we define $N_{ir}(t) = \sum_{j=1}^{m(t)} V_i(r, j)$. In particular, at any point in time $t$, $r$ maintains, for every $i \neq r$:

$$R_i(r) = \left(1 + \frac{F_i(t)}{F_r(t)}\right) N_{ir}(t).$$

Note that $F_i(t)$ and $N_{ir}(t)$ are available at $r$, while $F_r(t)$ is information carried and provided to $r$ by the visiting agent itself (also observe that, by definition, $N_{ir}(t) > 0$ implies $F_i(t) > 0$). The estimator above translates into the implementation of the $\text{UPDATE}(\cdot, \cdot)$ routine described in Figure 3, to dynamically update $R_i(r)$ at a generic node $r$ whenever a new agent $j$ visits the node. In particular, whenever agent some $j$ visits node $r$, $r$ initially updates its local counter (line 2), since it is receiving a new visit, while $j$ records in its history the number of agents that visited $r$ prior to its visit (line 3). This information will be provided by $j$ to nodes it visits in the sequel, if any. Finally (for cycle), for every item $i$ previously visited by $j$, $r$ updates $R_i(r)$ using the information carried by $j$.

The main result of this section is the proof that, over time and for every $i \neq r$, $R_i(r)$ provides an increasingly accurate estimation of the following function: $f(w_i) = w_i^2 \sum_{j=1}^{m(t)} p_k j$. The key point is that $f(w_i)$ is monotonically increasing in $w_i$ and thus can be used to rank items. This is stated in the following lemma, whose proof is straightforward and therefore omitted.

**Lemma 1:** For items $i$ and $h$ function $f$ verifies $f(w_i) \geq f(w_h)$ if and only if $w_i \geq w_h$.

With this choice of $f(\cdot)$, it is possible to prove that the following result holds, whose rigorous mathematical statement and proof are given in the appendix:

**Theorem 1:** If $i, r$ belong to the same cluster $C_k$ for some $k$, $R_i(r)$ becomes an increasingly accurate estimate of $f(w_i)$. In particular, accuracy becomes arbitrarily high as the number of users visiting $r$ increases over time.

**Convergence.** The result of Theorem 1 also describes the convergence properties of our algorithms. It is possible to prove that these bounds are asymptotically tight. A complete analysis is not the purpose of this paper. For the sake of completeness, we briefly address the simpler aspect of the estimation of $E[F_r(t)]$ at a generic node $r$. It is possible to prove the following theorem, whose rigorous mathematical statement and proof are omitted due to space limitations and will appear in the full version of the paper:

**Theorem 2:** Assume the cluster based model. For every $i \in C_k$ and for every $0 < \delta < 1$, $\Theta\left(\frac{\ln n}{m(t)}\right)$ are necessary and sufficient to estimate $E[F_r(t)]$ accurately.

Note that accurately estimating $E[F_r(t)]$ is crucial, since it appears in the explicit expression of $f(w_i)$.

\section{IV. Experimental analysis}

The experimental part of this paper focuses on the following aspects: i) assessing the soundness of the model of user behavior we consider; ii) assessing the effectiveness of recommendation algorithms. Both depend on the ability of our model to capture important trends in users' actual behavior.

**A. Model Validation**

We validate our assumptions analyzing the Netflix dataset on movies\footnote{Available at http://www.netflixprize.com/.}, consisting of one file for each movie. Each record of a movie file contains the id of a user, the rating of the movie given by the user and the date of the rating.

**Distribution of weights.** We considered 100 items uniformly sampled over the 770 available comedy movies in the Netflix dataset. The value of the $w_i$ for item $i$ in the comedy movies cluster is defined as the popularity of movie $i$, obtained by dividing the number of visits to $i$ by the total number of users.

We did not use users’ ratings directly to compute weights, since we did not observe a clear connection between the a priori choice of the movie based on its popularity and user’s taste and the a posteriori rating of the movie. As it can be observed in figure 4(a), our model fits actual user behavior reasonably well if we assume a Zipf’s distribution of the weights.

**Visit patterns.** As far as the order of visits is concerned, we calculated the correlation coefficient between the cluster weights and the order of visit on real data and we obtained a value of $-0.6$. This strongly supports the use of the weighted visit model. In contrast, figure 4(b) shows that the distribution of the number of visited items per user predicted by our model...
using a Zipf’s distribution for item weights differs with respect to real data. This is due to our simplifying assumption that users have the same profile within each cluster (though having different preferences for the same cluster). In particular, the peak observed in the synthetic distribution corresponds to the average visit length predicted by our model. This aspect could be taken into account by complicating the model, still our model seems to strike a good balance between accuracy and simplicity, since it seems able to predict important trends in user behavior, as shown in the following Subsection IV-C.

B. Performance Metrics

We evaluate the performance of the system along two main axes: the ability to accurately reconstruct the user profiles and the quality of recommendations. In particular, we evaluate the former in terms of ranking similarity and convergence speed, while the latter is evaluated in terms of standard measures of quality used in information retrieval [7], in particular hit ratio, precision and recall.

Ranking similarity. The quality of recommendation actually depends on the “distance” between the estimated and the real rankings. For this reason, we consider a standard measure of the distance between rankings, i.e., Kendall’s τ coefficient (Kendall’s τ for short). Kendall’s τ measures the degree of similarity between two rankings and it is defined as $\tau = 4P/(n(n-1)) - 1$, where $P$ is the sum, over all the items, of items ranked after the given item by both rankings (i.e. actual ranking and estimated one). Kendall’s τ enjoys the following properties: i) if the agreement between the two rankings is perfect (i.e., the two rankings are the same) the coefficient has value 1; ii) if the disagreement between the two rankings is perfect (i.e., one ranking is the reverse of the other) the coefficient has value -1; iii) for all other arrangements the value lies between -1 and 1, and increasing values imply increasing agreement between the rankings.

As an example, consider a set $\{A, B, C, D, E\}$ of 5 objects and the following rankings: $R_1 = (A, B, C, D, E)$ and $R_2 = (B, A, C, D, E)$. Clearly, the two rankings are very similar and indeed, applying the above definition, we obtain $\tau = 0.8$. On the other hand, if we consider ranking $R_3 = (E, C, D, A, B)$, $R_1$ and $R_3$ are “almost” inverted and this is reflected by their Kendall’s τ, which is $-0.6$ in this case.

Convergence Time. By convergence time we mean the number of visiting users before the system eventually produces an accurate estimation of the performance index under consideration.

HitRatio($\hat{T}$). Hit ratio measures the fraction of good recommendations provided by the items. According to figure 2, an item recommends $\hat{T}$ top items in $R(i)$ not present in $H(j)$ and there is a hit if at least one of the recommended items will be eventually visited by the user. HitRatio($\hat{T}$) is defined as the ratio between the number of hits and $\hat{T}$.

Precision($\hat{T}$) and Recall($\hat{T}$). Precision and recall are standard measure of the accuracy in providing relevant documents. Define by $D$ the set of items (i.e. the corpus) and by $D_j$ the set of relevant items for user $j$ and let $(d_1, d_2, ..., d_i)$ be the recommendations provided by the visited item and $(r_1, r_2, ..., r_i)$ where $r_i = 1$ if $d_i \in D_j$ and 0 otherwise. Then:

\[
\text{recall} = \frac{r_i}{\sum_{1 \leq i \leq \hat{T}} r_i}.
\]

I.e., recall is the fraction of all relevant items included in the recommendation. Furthermore:

\[
\text{precision} = \frac{1}{\hat{T}} \sum_{1 \leq i \leq \hat{T}} r_i.
\]

I.e., precision is the fraction of the top $k$ recommendations that are actually relevant. In the following we assume that $D_3$ is the set of items that will be visited in the future by a user.

C. Quality of recommendation

We evaluated the performance of our recommendation algorithm on both real Netflix and synthetic data. Each performance index has been computed by averaging the results over 10 independent runs with 100 items and 200000 users. As to Kendall’s τ, in each run it is first computed for each item whose summary contains at least two entries and then averaged over all such items. To generate synthetic inputs, we assume the weighted visit model with item weights within a cluster distributed according to Zipf’s law; since we are considering only comedy movies, we can assume that $p_{kj}$, the cluster preference, is one for all users $j$.

As a benchmark to evaluate the quality of our recommendation algorithm (alg) when considering hit ratio, precision and recall, we compare its performance to those of a random recommendation algorithm (rnd), namely the algorithm whose recommended item is randomly chosen. This allows us to measure the impact on performance resulting from our user profiling techniques.

Ranking similarity. Our first goal is to evaluate the ranking similarity between the actual cluster weights and the estimated ones by means of the Kendall’s τ coefficient (KT). We observe that the value of KT is very good on real data: with only 500 users its value is about 0.75. KT rapidly arrives to 0.84 with 8000 users (see figure 5) and then it slowly increases to about 0.87 with 200000 users. This behavior is confirmed and actually improved on synthetic data. We observe that KT tends to one confirming that the estimated ranking rapidly converges to the real one.

Hit Ratio. On real data, performance of alg is always sensibly better than rnd. In particular, HitRatio(1) of alg is more than 3 times better than rnd on real data. On synthetic data, this factor is 4 for $\hat{T} = 1$, it goes down to 2 for $\hat{T} = 4$. This can be explained by the fact that the hit ratio of rnd does not depend on the popularity of the items. On the contrary alg tries always to recommend the most popular items, but as $\hat{T}$ increases as successive items to be recommended are less popular and thus the likelihood of hits decreases. The higher quality of recommendations of our algorithm on real data can be explained considering the average length of the number of visited items per users (av). In fact the av on real data is about 7 while in synthetic data is about the half. Since the probability of a hit for a user clearly also depends on the
number of visits of the user, it follows that the higher is the $alv$ the higher is the hit ratio (similar considerations can be made for precision and recall).

**Precision and Recall.** Figure 6 shows the Precision and Recall of $alg$ and $rnd$ for different values of $\hat{T}$, both on real and synthetic data. Also in this case $alg$ out-performs $rnd$. The precision of $rnd$ is constant and does not depend on the number of suggestions provided (i.e. $\hat{T}$), as it can be easily proved considering that the random recommendation process is governed by an hypergeometric distribution. The precision of $alg$ tends to decrease as the number of recommendation increases; this is expected since, as we observed previously, the accuracy of recommendations depends on the popularity of the items and increasing the number of recommended items forces the algorithm to choose items of decreasing popularity, so more unlikely to meet user expectation. The much better performance on real data than synthetic ones can be explained by the same arguments as for the hit ratio.

Summarizing, the precision of $rnd$ is constant while the precision of $alg$ decreases as $\hat{T}$ increases and this is a further explanation of the reduction of the hit ratio gap observed before. Similar considerations apply to recall.

We briefly report that we also evaluated the performance of our algorithm when the random visit model is adopted. In this case, values for Kendall’s $\tau$ are sensibly lower; it is about 0.6 with 200000 users, a value that guarantees also in this case good performance on all the considered metrics. We observed in general a worsening in performance indices with respect to the weighted visit order model; this behavior is more evident as $\hat{T}$ increases.

**V. CONCLUDING REMARKS AND OPEN PROBLEMS**

This paper is a first step and leaves many open questions. We have considered the model in which items recommend other items within the cluster they belong to. While this is interesting in many realistic scenarios, an obvious extension is to consider more general models.

Another aspect concerns storage requirements and accuracy. In fact, maintaining at every node a summary with one entry for every other item visited in the past by users that also visited the item under consideration may in some applications be too expensive in terms of memory. These considerations naturally suggest the use of streaming techniques [19] to maintain item summaries.

Moreover, it is clear that our model entails an implicit communication among nodes/items, mediated by users. The point here is that we are not assuming the existence of a communication infrastructure. On one hand, this may be the case in some applications of interest, e.g., based on the use of RFIDs. On the other hand, it is interesting to investigate the
potential benefit achievable if limited communication among nodes is possible.

The effectiveness of spectral techniques in the general case, especially based on SVD or similar approaches has been known for many years [9], [6], [16] in the centralized case. Extending our approach to devise effective, fully decentralized estimators of spectral metrics is a challenging task.

REFERENCES

In the sequel, we denote by $S(j)$ the set of items visited by user $j$ during its permanence in the system. If $l \leq |S(j)|$, $S_{<l}(j)$ denotes the subset of the first $l - 1$ items visited by $j$. The following lemma holds:

**Lemma 3:** For every $S$ such that $\{i, r\} \subseteq S$, with $i, r \in C_k$ for some $k$, for every $j$:

$$P[V_{ir}(j) = 1 | S(j) = S] = \frac{w_i}{w_i + w_r}.$$  

**Proof:** Denote by $Y_{ir}(j)$ the item visited at the $l$-th step of $j$’s visit, where $Y_{ir}(j) = \emptyset$ if $l > |S|$. We have:

$$P[Y_{ir}(j) = i | S(j) = S] = \sum_{l=1}^{S_{<l}(j) \cap \{i, r\} = \emptyset} P[Y_{ir}(j) = i | S(j) = S \land (S_{<l}(j) \cap \{i, r\} = \emptyset)].$$

- $P[S_{<l}(j) \cap \{i, r\} = \emptyset | S(j) = S] \leq |S_{<l}(j)| = 1$.

where the first equality follows since, given $S(j) = S$, with $\{i, r\} \subseteq S$, $V_{ir}(j) = 1$ is equivalent to stating that $i$ is visited at some step where $r$ has not been visited yet. On the other hand, denote by $S(i, j, S)$ the set of all subsets of $S$ that $i$ contain $l - 1$ elements and it do not contain $\{i, j\}$. We have:

$$P[Y_{ir}(j) = i | (S(j) = S) \land (S_{<l}(j) \cap \{i, r\} = \emptyset)] = \sum_{W \in S_{i, j, S}} P[Y_{ir}(j) = i | (S(j) = S) \land (S_{<l}(j) \cap \{i, r\} = \emptyset)].$$

- $P[S_{<l}(j) \cap \{i, r\} = \emptyset | S(j) = S] = \frac{w_{ij}}{1 - \sum_{f \in W} w_{fj}}$.

by the definition of the weighted visit process described above.

Analogously:

$$P[Y_{ir}(j) = r | (S(j) = S) \land (S_{<l}(j) = W)] = \frac{w_{rj}}{1 - \sum_{f \in W} w_{fj}},$$

and

$$P[Y_{ir}(j) = r | (S(j) = S) \land (S_{<l}(j) \cap \{i, r\} = \emptyset)] = \sum_{W \in S_{i, j, S}} P[Y_{ir}(j) = r | (S(j) = S) \land (S_{<l}(j) \cap \{i, r\} = \emptyset)].$$

This implies that the expressions of $P[V_{ir}(j) = 1 | S(j) = S]$ and $P[V_{ir}(j) = 1 | S(j) = S]$ are the same up to multiplying factors, which are $w_{ij} = p_{kj}w_i$ and $w_{rj} = p_{kj}w_r$ respectively. Therefore:

$$P[V_{ir}(j) = 1 | S(j) = S] = \frac{w_i}{w_i + w_r}$$

**Proof:** We obviously have $F(t) = \sum_{j=1}^{m(t)} V_{ij}(j)$ and $P[V_{ir}(j) = 1] = p_{kj}w_i$. This immediately gives $E[F(t)] = w_i \sum_{j=1}^{m(t)} p_{kj}$. Furthermore, agents visits are independent of each other. Hence, applying Chernoff bound to $\sum_{j=1}^{m(t)} V_{ij}(j)$ [18] yields the result.

---

**APPENDIX**

In the sequel, we set $\bar{f}_i(t) = R_i(r)$, the estimate of $f(w_i)$ maintained at time $t$ at node $r$. The proof of Theorem 1 is implied by proving the following statement:

If $i, r \in C_k$ for some $k$:

$$f(w_i) = \left(1 + \frac{1}{\delta} \frac{E[F_i(t)]}{E[F_i(t)]} \right) E[N_{ir}(t)].$$

Furthermore, for every $r$ and $i$ belonging to the same cluster $C_k$, whenever $t$ is large enough that $\sum_{j=1}^{m(t)} p_{kj} \geq \frac{3(w_i + w_r)}{2r^2 w_{ir}} \ln \frac{6}{\epsilon}$ with $\epsilon \leq 1/5$;

$$P[1 - 3\epsilon]f(w_i) \leq f_i(t) \leq (1 + 4\epsilon)f(w_i) \geq 1 - \delta.$$  

We first give the following Lemma that will be useful later:

**Lemma 2:** For every $i \in C_k$: $E[F_i(t)] = w_i \sum_{j=1}^{m(t)} p_{kj}$. Furthermore, for every $\delta, \epsilon > 0$, as soon as $t$ is such that $\sum_{j=1}^{m(t)} p_{kj} \geq \frac{3}{\epsilon^2 w_{i}} \ln \frac{2}{\delta}$

$$P[F_i(t) - E[F_i(t)] > \epsilon E[F_i(t)] \leq \delta.$$  

**Proof:** We obviously have $F_i(t) = \sum_{j=1}^{m(t)} V_{ij}(j)$ and $P[V_{ir}(j) = 1] = p_{kj}w_i$. This immediately gives $E[F_i(t)] = w_i \sum_{j=1}^{m(t)} p_{kj}$. Furthermore, agents visits are independent of each other. Hence, applying Chernoff bound to $\sum_{j=1}^{m(t)} V_{ij}(j)$ [18] yields the result.
Notice that, at every node $r$ and at any time $t$, we are in fact observing the variable $N_{ir}(t) = \sum_{j=1}^{m(t)} V_{ir}(j)$. As to $P[V_{ir}(j) = 1 | r \in S(j)]$, we have:

**Lemma 4:** If $i, r \in C_k$: $P[V_{ir}(j) = 1] = \frac{p_{kj} w_i^2}{w_i + w_r}$.

**Proof:** We have:

$$P[V_{ir}(j) = 1 | r \in S(j)] = \sum_{S \subseteq S} P[V_{ir}(j) = 1 | S(j) = S] P[S(j) = S | r \in S(j)]$$

$$= \sum_{S \subseteq S} P[V_{ir}(j) = 1 | S(j) = S] P[S(j) = S | r \in S(j)]$$

$$= \frac{w_i}{w_i + w_r} \sum_{S \subseteq S} P[S(j) = S | r \in S(j)]$$

$$= \frac{w_i}{w_i + w_r} P\{i, r \subseteq S(j) | r \in S(j)\} = \frac{p_{kj} w_i^2}{w_i + w_r},$$

where the second inequality follows since $V_{ir}(j) = 0$ deterministically if $i \not\in S(j)$, the third follows from Lemma 3 and the fifth follows since the events $(i \in S(j))$ and $(r \in S(j))$ are statistically independent. The claim then follows since $P[r \in S(j)] = p_{kj} w_r$.

**Theorem 3:** If $i, r \in C_k$: $E[N_{ir}(t)] = \frac{w_i^2 w_r}{w_i + w_r} \sum_{j=1}^{m(t)} p_{kj}$.

Furthermore: $P[|N_{ir}(t) - E[N_{ir}(t)]| > \epsilon E[N_{ir}(t)]] \leq \delta$, as soon as $t$ is large enough that $\sum_{j=1}^{m(t)} p_{kj} \geq \frac{3(w_i + w_r)^2}{\epsilon^2 w_i^2 w_r} \ln \frac{2}{\delta}$.

**Proof:** The first claim follows immediately from Lemma 4. The second claim follows from a simple application of Chernoff bound [18] to the variable $N_{ir}(t)$.

In order to analyze the accuracy of the estimator above and the rationale behind, we assume that agents visit the system one at a time. We show in the full paper how this assumption can easily be removed. This assumption implies that, if agent $j$ visits at time $t_1$ and $r$ at time $t_2 > t_1$, we have $m(t_1) = m(t_2)$. Furthermore, $E[F_i(t_2)] = E[F_i(t_1)] = w_i m(t_1)$ and $E[F_r(t_2)] = w_r m(t_1)$. The following holds:

**Lemma 5:** If at most 1 agent visits the system at any time $t$ and $i, r \in C_k$ for some $k$:

$$f(w_i) = w_i^2 \sum_{j=1}^{m(t)} p_{kj} = \left(1 + \frac{E[F_i(t)]}{E[F_i(t)]}\right) E[N_{ir}(t)].$$

**Proof:** The proof follows immediately, observing that $E[F_i(t)] / E[F_i(t)] = w_i / w_i$ from Lemma 2 and substituting $w_r = w_i E[F_i(t)] / E[F_i(t)]$ in the expression of $E[N_{ir}(t)]$ in Theorem 3.

Lemma 5 shows that the estimator we are using is in fact a simple plug-in estimator for $f(w_i)$. We can finally prove the claim of the theorem, i.e., that the approximation of $f(w_i)$ becomes more and more accurate over time.

In the sequel of this proof we drop $t$ from the notation, since it is understood from context. We also recall that $F_i$, $F_r$ and $N_{ir}$ are each the sum of binary independent variables by the independence of the agents’ visits. Hence, if $\sum_{j=1}^{m(t)} p_{kj} \geq \frac{3(w_i + w_r)}{\epsilon^2 w_i^2 w_r} \ln \frac{6}{\delta}$, simple applications of Lemma 2 and Theorem 3 allow to conclude that each of the following events occurs with probability at most $\delta/3$: i) $|F_i - E[F_i]| > \epsilon E[F_i]$; ii) $|F_r - E[F_r]| > \epsilon E[F_r]$; iii) $|N_{ir} - E[N_{ir}]| > \epsilon E[N_{ir}]$. Hence, with probability at least $1 - \delta$ we have:

$$T_i \leq \left(1 + \frac{1 + \epsilon}{1 - \epsilon} E[F_i] \right) (1 + \epsilon) E[N_{ir}]$$

$$(1 + 6\epsilon) \left(1 + \frac{E[F_i(t)]}{E[F_i(t)]}\right) E[N_{ir}(t)],$$

where the first inequality follows since we have $N_{ir} \leq (1 + \epsilon)E[N_{ir}]$, $F_i \leq (1 + \epsilon)E[F_i]$ and $F_r \geq (1 - \epsilon)E[F_r]$, while the second inequality holds if $\epsilon \leq 1/5$. Analogously:

$$T_i \geq \left(1 + \frac{1 - \epsilon}{1 + \epsilon} E[F_i] \right) (1 - \epsilon) E[N_{ir}]$$

$$(1 - 3\epsilon) \left(1 + \frac{E[F_i(t)]}{E[F_i(t)]}\right) E[N_{ir}(t)],$$

where the first inequality follows from a similar argument as above, while the second inequality follows from trivial manipulations. Recalling Lemma 5 we complete the proof of Theorem 1.
Data Stream Management Systems for Processing and Mining RFID Streams

Carlo Zaniolo
UCLA Computer Science Department
Los Angeles California 90095
Email: zaniolo@cs.ucla.edu

Abstract—RFID technology offers significant advantages over traditional object-tracking technology and is increasingly adopted and deployed in many applications. Such applications generate large volumes of streaming data, which have to be filtered, processed, and transformed into higher-level information for integration into business applications. For instance, RFID data are highly temporal and form diverse patterns of complex temporal events that must be detected and managed in different applications. This situation calls for the development of powerful Data Stream Management Systems (DSMS) that can support complex event processing of massive RFID data streams with Quality of Service (QoS) guarantees. Such DSMS must enable applications ranging from low-level data cleaning, to complex temporal event pattern detection and sophisticated data mining on RFID data streams. This talk will (i) describe how complex event detection and management can be achieved through Kleene-closure extensions of continuous query languages, and (ii) show that data stream mining can be achieved through user-defined aggregates written in the same language. This advanced functionality is realized efficiently in Stream Mill—a DSMS designed for power and extensibility that provides a powerful platform for processing RFID information.

I. INTRODUCTION

Applications that use massive data streams generated by RFID and other sensors require computing technology that can process such data efficiently and reliably, even when real-time response is required. Thus, RFID represent a natural application area for Data Stream Management Systems (DSMS) [1] and Complex Event Processing (CEP) systems, and several startup companies are currently marketing their DSMS and CEP systems for RFID applications. Because of the obfuscation caused by aggressive marketing, the DSMS and CEP systems are often viewed as synonyms in the marketing arena, but they actually build on very different technologies. DSMS focus on supporting continuous queries on data streams with QoS guarantees. Thus, their technical approach consists in (i) adapting and extending Database query languages, such as SQL and XQuery, to express continuous queries on the incoming streams, and (ii) advanced techniques for scheduling, synopses, and load shedding to achieve efficiency, scalability and online response. On the other hand CEP systems seek to support a more complex poset of events, called event clouds [2], by building on platform such as Event-Condition-Action rules, message brokers, and special-purpose programming languages [2]. Therefore, CEP systems seek to support applications of greater diversity and complexity than DSMS, which instead are optimized for processing streams of temporally ordered events. Thus, while DSMS might achieve better performance, scalability, in the number of queries running and the event arrival rate, the generality of their data stream models and continuous query languages are more limited. In particular the expressive power limitations that SQL experience on databases, are further exacerbated on data streams by the exclusion of blocking query operators and the difficulty of embedding continuous queries into procedural programming languages [3].

To overcome these limitations we have developed Stream Mill, a DSMS prototype designed for power and generality [6], which can support the whole range of applications associated with RFID—from basic ones such as data cleaning and filtering [4], to the more complex ones, such as trajectory identification and data stream mining. For instance, RFIDs are often used to track moving objects that generate spatio-temporal trajectories. These can be identified through Kleene-closure extensions of database query languages which were supported in Stream Mill and now proposed for inclusion in SQL standards [5]. At the high-end of the spectrum, we find data stream mining which must contend with both the complexity of online mining algorithms and the limited expressive power of continuous query languages. SMM is an extension of the Stream Mill system that solves this problem through user-defined aggregates, and mining methods [7].

ACKNOWLEDGMENT

The author would like to thank his students past and present for their important contribution to Stream Mill.

REFERENCES
