Practical Data Prediction for Real-World Wireless Sensor Networks

Usman Raza, Alessandro Camerra, Amy L. Murphy, Themis Palpanas, and Gian Pietro Picco

Abstract—Data prediction is proposed in wireless sensor networks (WSNs) to extend the system lifetime by enabling the sink to determine the data sampled, within some accuracy bounds, with only minimal communication from source nodes. Several theoretical studies clearly demonstrate the tremendous potential of this approach, able to suppress the vast majority of data reports at the source nodes. Nevertheless, the techniques employed are relatively complex, and their feasibility on resource-scarce WSN devices is often not ascertained. More generally, the literature lacks reports from real-world deployments, quantifying the overall system-wide lifetime improvements determined by the interplay of data prediction with the underlying network. These two aspects, feasibility and system-wide gains, are key in determining the *practical* usefulness of data prediction in real-world WSN applications.
In this paper, we describe Derivative-Based Prediction (DBP), a novel data prediction technique much simpler than those found in the literature. Evaluation with real data sets from diverse WSN deployments shows that DBP often performs better than the competition, with data suppression rates up to 99% and good prediction accuracy. However, experiments with a real WSN in a road tunnel show that, when the network stack is taken into consideration, DBP only *triples* lifetime—a remarkable result per se, but a far cry from the data suppression rates above. To fully achieve the energy savings enabled by data prediction, the data and network layers must be jointly optimized. In our testbed experiments, a simple tuning of the MAC and routing stack, taking into account the operation of DBP, yields a remarkable *seven-fold* lifetime improvement w.r.t. the mainstream periodic reporting.

Index Terms—Wireless sensor networks, data prediction, time series forecasting, energy efficiency, network protocols

1 Introduction

Wireless sensor networks (WSNs) provide the flexibility of untethered sensing, but pose the challenge of achieving long lifetime with a limited energy budget, often provided by batteries. It is well-known that communication is the primary energy drain, which is unfortunate, given that the ability to report sensed data motivates the use of WSNs in several pervasive computing applications.

An approach to reduce communication without compromising data quality is to *predict* the trend followed by the data being sensed, an idea at the core of many techniques [1]. This data prediction approach[1] is applicable when data is reported periodically—the common case in many pervasive computing applications. In these cases, a model of the data trend can be computed locally to a node. This model constitutes the information being reported to the data collection sink, replacing several raw samples. As long as the locally-sensed data are compatible with the model prediction, no further communication is needed: only when the sensed data deviates from the model, must the latter be updated and sent to the sink. Section 2 formulates the data prediction problem in more detail.

The aforementioned approach is well-known, and has been proposed by several works we concisely survey in Section 6. Nevertheless, to the best of our knowledge none of these works has been verified in practice, in a real-world WSN deployment. On one hand, the techniques employed are relatively complex, and their effectiveness is typically evaluated based on implementations in high-level languages (e.g., Java) on mainstream hardware platforms. Therefore, their feasibility on resource-scarce WSN devices remains unascertained. Moreover, the works in the literature typically evaluate the gains only in terms of messages suppressed w.r.t. a standard approach sending all samples. This data-centric view, however, is quite optimistic. WSNs consume energy not only when transmitting and receiving data, but also in several *continuous* control operations driven by the network layer protocols, e.g., when maintaining a routing tree for data collection, or probing for ongoing communication at the MAC layer.

Therefore, the true question, currently unanswered by the literature, is to what extent the theoretical savings enabled by data prediction are actually observable *in practice*, i.e., *i)* on the resource-scarce devices typical of WSNs, and *ii)* when the application and network stacks are combined in a single, deployed system. The goal of this paper is to provide an answer to this question, through the following contributions:

- We propose *Derivative-Based Prediction* (DBP), a novel data prediction technique compatible with applications requiring hard guarantees on data quality. DBP, described in Section 3, predicts the trend of data measured by a sensor node, and is considerably simpler than existing methods, making it amenable for resource-scarce

- *U. Raza is with the Bruno Kessler Foundation, Italy and the University of Trento, Italy. E-mail: raza@fbk.eu*
- *A. Camerra is with IBM, Italy. E-mail: alessandro.camerra@it.ibm.com*
- *A.L. Murphy is with the Bruno Kessler Foundation, Italy. E-mail: murphy@fbk.eu*
- *T. Palpanas is with the Paris Descartes University, France and the University of Trento, Italy. E-mail: themis@mi.parisdescartes.fr*
- *G.P. Picco is with the University of Trento, Italy. E-mail: gianpietro.picco@unitn.it.*

1. The techniques discussed here are known under various names, including *time-series forecasting*, *data modeling*, *prediction-based data collection*, and *model-driven data acquisition*. Although in a preliminary version of this paper [2] we used the last term, in this paper we resort to the more intuitive *data prediction*.

WSNs, as witnessed by our TinyOS implementation for the popular TelosB motes [3].

- We perform an extensive experimental evaluation of DBP against state-of-the-art data prediction techniques, based on 7 diverse real-world data sets with more than 13 million data points in total. The results demonstrate the effectiveness of DBP, which often performs better than the competition by suppressing up to 99% of data transmissions while maintaining data quality within the required application tolerances.
- We describe the first[2] study of the interaction of data prediction with WSN network protocols, directly comparing the theoretical application-level gains against the practical, system-wide ones. We evaluate the performance of a staple network stack consisting of CTP [4] and Box-MAC [5], both in an indoor testbed and a real application setting, a road tunnel [6]. Our results show that the gains attained in practice lead to three- to five-fold WSN lifetime improvements, which is a significant achievement in absolute terms, but dramatically lower than those derived in theory.
- We explore the potential of cross-layer network stack optimizations to further improve the lifetime of WSN nodes running DBP. In our tunnel application, we show how a careful, yet simple, joint parameter tuning of the MAC and routing layers reduces the network control overhead considerably, without affecting the DBP operation, and yields a remarkable *seven-fold* lifetime improvement w.r.t. the standard periodic reporting.

The paper ends with the concluding remarks of Section 7, underlining the further lifetime improvements and enhanced reliability that can be attained by a WSN network stack expressly designed to work in conjunction with data prediction techniques.

2 Problem Formulation

Data collection is a fundamental functionality of many WSN applications, and is commonly implemented by nodes periodically taking sensor measurements and reporting the corresponding samples to a data sink.

The premise of applying data prediction is that communication can be significantly reduced by avoiding transmission of each raw sample to the sink. This is achieved by using a model to estimate the sensed values, and by communicating with the sink only when changes in the sampled data render the model no longer able to accurately describe them.

The data prediction strategy, applied on each node, involves the following general steps. The sensor node builds a model of its data based on some initial, observed values, and transmits the model to the sink. From that point on, the sink operates on the assumption that the data observed by the sensor node are within the value tolerance of the data predicted by the model. At the same time, the node is also using the model to predict its own sensor data, and compares the predicted values with those actually observed. If their difference is within the error tolerance, no further action is required. Otherwise, the sensor node builds a new model and transmits it to the sink.

2. A preliminary version of this paper appeared in [2].

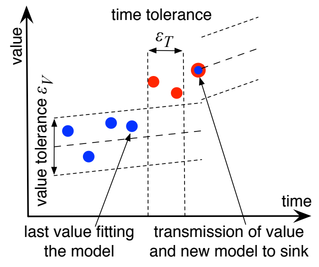

time tolerance
ε_T
value
value tolerance ε_V
time
last value fitting the model
transmission of value and new model to sink

Fig. 1. Value and time tolerance.

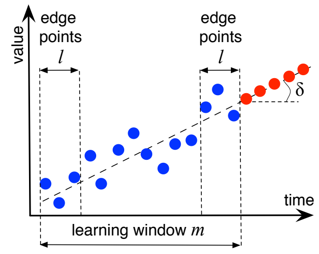


Fig. 2. Derivative-based Prediction.

To enable this strategy, the application running at the sink must allow for a small tolerance in the accuracy of the reported data—an assumption that holds in the majority of WSN applications. In contrast with the ideal requirement of the sink obtaining *exact* values in *all* data reports, the correctness of these applications is unaffected as long as

1) the reported values match *closely* the exact ones;
2) inaccurate values occur only *occasionally*.

In other words, deviations from the exact reports are acceptable, as long as their extent in terms of difference in *value* and *time interval* during which the deviation occurs are small enough. In this paper we only consider non-probabilistic techniques that can provide *hard guarantees* on their predictions, a requirement for several real-world applications. We capture these assumptions with the following definitions:

- Let V_i be an exact measurement taken at time t_i. The *value tolerance* is defined by the maximum relative and absolute errors acceptable, $\varepsilon_V = (\epsilon^{rel}, \epsilon^{abs})$. From the application perspective, reading a value V_i becomes equivalent to reading any value $\hat{V}_i$ in the range R_V defined by the maximum error, $\hat{V}_i \in R_V = [V_i - \epsilon, V_i + \epsilon]$, where $\epsilon = \max\{\frac{V_i}{100}\epsilon^{rel}, \epsilon^{abs}\}$. In other words, the application considers a value $\hat{V}_i \in R_V$ as *correct*.
- Let $T = |t_j - t_k|$ be a time interval, and $\hat{V}_T = \{\hat{V}_j, \ldots, \hat{V}_k\}$ the set of values reported to the application during T. The *time tolerance* ε_T is the maximum acceptable value of T such that *all* the values reported in this interval are incorrect, i.e., $\hat{V}_i \notin R_V, \forall \hat{V}_i \in \hat{V}_T$.

The intuition behind these is shown in Fig. 1. Data prediction aims to suppress as many data reports from the WSN nodes as possible, while ensuring that the data used by the application at the sink is within the value and time tolerances ε_V and ε_T specified as part of the requirements. The use of both absolute and relative errors in the value tolerance is dictated by the requirements of many applications in which values can be both very small and very large. Our evaluation in Section 4 shows a concrete example, the TUNNEL application, where this problem is evident. If only the absolute error ϵ^{abs} is used, it is difficult to set it in a way meaningful for both very small and very large values. On the other hand, a relative error ϵ^{rel} is often not very useful in the case of very small values, when the quantities at stake are negligible. Using the maximum between relative and absolute error as the value tolerance allows one to specify error in relative terms, and at the same time set an absolute threshold beyond which variations can be ignored.

3 Derivative-Based Prediction (DBP)

With this general statement of the problem, we next turn to the details of our solution as embodied in the novel data prediction technique we refer to as Derivative-Based Prediction (DBP). We first discuss the rationale and offer the intuition behind our algorithm, then present the technical details and implementation considerations.

Goals and Requirements. The idea behind DBP is to use a simple model that can effectively capture the main data trends, and to compute it in a way that is resilient to the noise inherent in the data. DBP is based on the observation that the trends of sensed values in short and medium time intervals can be accurately approximated using a linear model. Although this idea has already appeared in the literature, there is a key difference to our approach: previous studies compute models that aim to reduce the approximation error (i.e., Root Mean Squared Error, RMSE) w.r.t. recent *data points*, producing models that are accurate w.r.t. *past* data. Instead, DBP computes models capturing the *trends* in recently-observed data, producing models accurate w.r.t. *future* data. Evidently, these trends are best modeled when data exhibit short-term linear behavior. As we discuss in the experimental evaluation, DBP can effectively describe data with long-term non-linear trends, at the cost of more frequent model updates.

Technical Details. Fig. 2 provides an illustration of DBP. The model is computed based on a *learning window*, containing m data points; the first and the last l we call *edge points*. The model is linear, computed as the slope δ of the segment connecting the average values over the l edge points at the beginning and end of the learning window. This computation resembles the calculation of the derivative, hence the name *Derivative-Based Prediction*. Involving only $2l$ points makes DBP computationally efficient, and therefore appealing for implementation on resource-scarce nodes, but also robust against noise and outliers present in the samples, as shown in Section 4.2.

The first DBP model generated is sent to the sink. From that time on, each node buffers a sliding window of the last m data points (the learning window) sampled from its sensor. Upon sampling a point, the "true" value sensed is compared to the one "predicted" by DBP according to the current model, i.e., following the slope δ. If the sensor reading is within the value tolerance ε_V w.r.t. the model, no action is required: the sink automatically generates a new value that is an acceptable approximation of the real one. Otherwise, if the readings continuously deviate from the model for more than ε_T time units, a new model must be recomputed based on the m buffered data points. The model is then sent to the sink.

Implementation Considerations. As our final goal is to deploy DBP on real WSN nodes, the complexity and resource requirements (i.e., memory and CPU) of the implementation are very important, as these devices are typically not equipped with large memory or powerful CPUs. For instance, the popular TelosB motes used by the majority of WSN deployments reported in the literature, including the one about adaptive lighting in road tunnels [6] we illustrate in the next section, are equipped with only 48 kB of code memory, 10 kB of RAM, and an 8 MHz micro-controller suited for integer operations only.

In this respect, DBP is very efficient, involving only one subtraction, two summations, and two divisions to build the model, and a single summation for predicting the next value. Our DBP implementation in TinyOS requires only 50 lines of low-level code (equivalent to only 8 lines of Java code), without including any external libraries, or using floating point arithmetic. As node memory is limited, eliminating the floating point arithmetic module is highly desirable. Further, our DBP implementation uses only 108 B of RAM, leaving almost all of the data memory to the application and the network stack.

In contrast, other state-of-the-art techniques (e.g., those compared in Section 4) employ mathematical libraries for solving linear equations with 2–3 unknowns to compute an autoregressive model (SAF), and a linear (PLA and SAF) and quadratic polynomial (POR) regression using least squares minimization. Such requirements render these approaches considerably more resource-intensive. At the same time, DBP does not sacrifice accuracy, as shown in Section 4.2.

4 Application-level Evaluation

This section analyzes the ability of our data prediction technique, DBP, to reduce the amount of data that must be transmitted to the sink. This is notably different from the system-wide energy savings enabled by such data suppression, which we analyze in Section 5.

We evaluate and compare data prediction techniques using the *suppression ratio*

$$SR = 1 - \frac{\text{\# messages generated with prediction}}{\text{\# messages generated without prediction}}$$

as our primary performance metric. SR directly measures the fraction of application-layer messages whose reporting can be avoided: the higher the value of SR, the more effectively a technique is performing.

4.1 Applications and Datasets

Our evaluation is based on 7 datasets from 4 applications, described next, covering a variety of data variation patterns, sampling periods, and number of nodes. Table 1 outlines the main features of the datasets. Moreover, it reports the error tolerance we set as a requirement, based on the *real* one used in the application as defined by its designers or, in absence, by considering the nature of the application. Finally, we report the learning window m, which is a characteristic not only of DBP but of all approaches, and is set at the same value for the sake of comparison.

These datasets contain *real* collected data, which was subject to losses on the wireless channel or to hardware failures of some nodes. This is different from an online application of data prediction, where each node has a perfect record of the sensed values, as they are being sampled on the node itself. Therefore, before running our evaluation, we reconstruct a perfect data series for each node by removing

TABLE 1
Datasets characteristics and evaluation parameters

Application	Dataset	Sampling Period	Nodes	Samples	Error Tolerance (ϵ^{rel},ϵ^{abs})	ε_T	Learning Window (m)
TUNNEL	Light	$30s$	40	5,414,400	(5%, 25 counts)	2	20
SOIL	Air Temperature	10 minutes	10	225,360	(5%, 0.5°C)	2	6
	Soil Temperature	10 minutes	4	77,904	(5%, 0.5°C)	2	6
INDOOR	Humidity	$31s$	54	2,303,255	(5%, 1%)	2	20
	Light	$31s$	54	2,303,255	(5%, 15 lx)	2	20
	Temperature	$31s$	54	2,303,255	(5%, 0.5°C)	2	20
WATER	Chlorine	5 minutes	166	715,460	(5%, 0.0001)	2	6

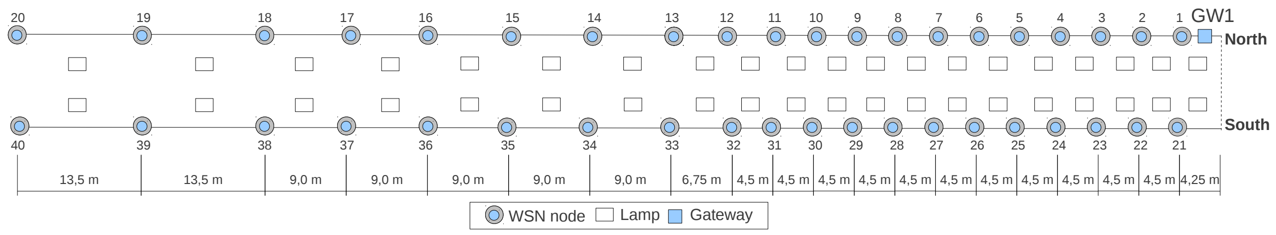


Fig. 3. Physical placement of WSN nodes in TUNNEL.

duplicates and interpolating for missing values, in the line of similar evaluations found in the literature [7].

Adaptive Lighting in Road Tunnels (TUNNEL). Our first case study involves a real-world WSN application, deployed in a road tunnel to acquire light readings [6]. The values are relayed in multi-hop to a gateway, and from there to a Programmable Logic Controller (PLC) that closes the control loop by setting the intensity of the lamps inside the tunnel. In contrast with the state of the art in tunnels, where light intensity is pre-set based on the current date and time, or at best determined by the external conditions, this closed-loop adaptive lighting system maintains optimal light levels by considering the *actual* conditions inside the tunnel. This increases safety, and enables considerable energy savings.

WSNs are an asset in this scenario, as nodes can be placed anywhere along the tunnel, not only where power and networking cables can reach. This drastically reduces installation and maintenance costs, and makes WSNs particularly appealing for existing tunnels, where infrastructure changes should be minimized. The downside to such flexibility is reliance on an autonomous energy source. Nevertheless, battery costs are minimal and their replacement can be combined with regularly-planned tunnel maintenance.

Fig. 3 shows the placement of WSN nodes inside our 260 m long, two-way, two-lane tunnel. Overall, 40 nodes are split evenly between the tunnel walls and placed at a height of 1.70 m, compatible with legal regulations. Their data reports are collected by a gateway, installed 2 m from the entrance. Each node is functionally equivalent to a TelosB mote [3], augmented with a sensor board equipped with 4 ISL29004 digital light (illuminance) sensors. This setup is similar to the one reported in [6], where we detail and evaluate the operational WSN-based, closed-loop adaptive lighting system. In this paper we use a different application and network stack, and compare data prediction techniques against the baseline represented by the aforementioned periodic reporting of all samples.

The dataset we use contains the light readings reported every 30 s from each node for 47 days, for a total of $5,414,400$ measurements—the largest among the datasets we consider here. To offer an intuition of the data, the dashed line in the top of Fig. 6(b) shows the raw sensor readings at a single node near the entrance of the tunnel over a one-day period. To establish the proper value and time tolerances, we consulted the lighting engineers who designed the control algorithm that establishes the lamp levels. By taking into consideration the inherent error of illuminance sensors, they determined a value tolerance $\varepsilon_V = (5, 25)$, i.e., values generated by the model can differ from the raw sensor reading by at most 5% or 25 counts, the latter corresponding approximately to 15 lx. Based on the application requirement that lamp levels must be adjusted slowly to minimize the effects of changes on the drivers, they also identified a time tolerance of one minute. For convenience, we express ε_T in terms of the 30 s reporting intervals of the application; a one-minute time tolerance corresponds to $\varepsilon_T = 2$. We further establish the number of values in the learning phase of data prediction techniques to be $m = 20$, corresponding to a period of 10 minutes.

Soil Ecology (SOIL). Our second case study uses data originating in the Life Under Your Feet (LUYF) project [8]. LUYF brings biologists and WSN experts together to study the soil micro-climate in different forests of Maryland. As environmental conditions affect the activities and behavior of plants, micro-organisms and insects in the soil, a large WSN offers accurate, fine-grained spatial and temporal data, collected without being intrusive to the living creatures. Our study uses the soil and air temperature datasets collected in an urban forest in Baltimore, over a period of 225 days between September 2005 and July 2006. The soil and air temperatures are measured on the surface of earth and inside the box of the node, respectively. Despite their commonalities such as the presence of a diurnal cycle, the two temperature datasets exhibit distinctly different variation patterns. The soil temperature varies gradually over time with changes lagging behind those of the air temperature by several hours due to the large inertia caused by the soil. We determined the value tolerance for the temperature datasets in consultation with the soil scientists of LUYF project. Interestingly, the scientists are not interested in the temperature itself, rather in the production of CO_2 due to the respiration of organisms and plants in soil, which is affected by temperature. A significant change in the concentration of CO_2 occurs with temperature changes of $0.5°$ C or 5% of the actual temperature. Given the sampling period of 10 minutes, we set the learning phase to 1 hour to accumulate $m = 6$ samples before applying the prediction technique.

Indoor Sensing (INDOOR). Our next application case study is arguably one of the first publicly available datasets collected from a WSN. As such it has been used by earlier data prediction studies [7], [9], offering direct comparison between our results and prior published results. In this dataset, the light, temperature, humidity, and battery voltage of 54 nodes (Mica2Dot) deployed inside the Intel Berkeley Research Lab are collected. The data trends are dramatically different from those of outdoor WSNs, as the indoor sensors are influenced by artificial factors such as the heating ventilation and air conditioning (HVAC) system and human-controlled lighting.

The dataset covers 36 days, in which the nodes reported 2.3 million values for each of the aforementioned physical quantities. In our experiments, we do not consider voltage as it is highly correlated to temperature. With this dataset, we set the tolerance parameters for temperature as in SOIL, and those of the other two quantities as an estimate of the perceivable effect on the comfort of the building occupants.

Water distribution (WATER). In our final case study we consider a simulated sensor network monitoring hydraulic and chemical phenomena in drinking water distribution piping systems. The data comes from EPANET 2.0 [10], an accurate modeling tool that tracks the water flow in each pipe, the water height in each tank, the pressure at each node, and the chlorine concentration throughout the network during a specified simulation period. This dataset has been used in several previous studies (e.g., in [11]–[14]) and thus offers a valuable point for comparative studies.

From this application, we consider a dataset containing measurements of the chlorine concentration every 5′ at 166 junctions in the water distribution network for a 15-day interval, for a total of $715,460$ measurements. This dataset exhibits a global, daily periodic pattern following residential demand, partly shown in Fig. 6(b), with a slight time shift across different junctions, due to the time it takes for fresh water to flow down the pipes from the reservoirs. We assume a value tolerance of (5, 0.0001), which allows sensors measuring very low chlorine concentrations to report data. In general, the data exhibits a periodic, quasi-sinusoidal pattern whose frequency is higher than in our other datasets. As such, it is more difficult to model with linear prediction techniques, and thus constitutes a worst-case scenario for our evaluation.

4.2 Comparing DBP against the State of the Art

The goal of data prediction is to reduce the transmission ratio without crossing the tolerated error values. To evaluate this, we consider all the available data sets described earlier and compare the suppression percentage of DBP to several other techniques from the literature we concisely describe here, and place in a wider context in Section 6:

- *Piecewise Linear Approximation* (PLA) is a popular technique that uses least square error linear segments to approximate a set of values [9]. In our case, each node uses a single segment to model sensed values.
- *Similarity-based Adaptable Framework* (SAF) [7] relies on an autoregressive moving-average model of order 3 with a moving-average parameter of order 0. In SAF a value V_i is predicted by a linear combination of the last three: $V_i = l_i + \alpha_1(V_{i-1} - l_{i-1}) + \alpha_2(V_{i-2} - l_{i-2}) + \alpha_3(V_{i-3} - l_{i-3})$, where α_1, α_2, α_3 are constants the model must estimate, and l_i models the linear trend of data over time.
- *Polynomial Regression* (POR). In contrast to DBP, POR allows the use of non-linear models for prediction. Intuitively, this may yield better performance through a better fit to the data. Like PLA, POR uses the least squares measure for selecting the most appropriate coefficients for the polynomials, which have the form $y = \sum_{i=0}^{p} \alpha_i x^i$. For this study, we evaluated polynomials of order $p = 2$, 3, 4, but show only $p = 2$ as it provides the best results for POR.

We used the value and error tolerances matching each target application as outlined in Table 1. The duration of the learning window m is the same across all techniques, and is also specified in Table 1. Finally, for DBP we used $l = 3$ edge points; this value yields the best performance, although its impact is nonetheless rather limited, as we show in Section 4.4.

First we consider the error of predicted vs. actual sensor values. Like other studies [9], [11], we use Root Mean Squared Error (RMSE) as an indicator of the quality of the predicted time series at the sink. We define it as $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(V_i - \hat{V_i})^2}$ where V_i and $\hat{V_i}$ are the sensed and predicted values, respectively, and N is the total number of values sensed. Table 2 shows the RMSE across all data sets. DBP minimizes the error in 4 of 7 datasets and is second best in the others, confirming its ability to accurately predict the sensed values. This impressive result may be surprising, considering that the approaches we compare against, unlike DBP, are expressly designed to reduce RMSE. Note however, that these techniques try to reduce the RMSE between the produced model and the *past* data values, while here we are interested in measuring the RMSE between the model and the *future* data values.

All approaches perform well in terms of data suppression, but DBP achieves the best results in 5 out 7 datasets. Table 2 shows that DBP suppresses 99.7% of the message reports in TUNNEL, and 99.6% and 99.5% for the temperature and humidity INDOOR datasets.

On the other hand, the WATER dataset is characterized by non-linear periodic trends, that are better approximated by the polynomial regression function of POR rather than

TABLE 2
Root Mean Squared Error and Suppression Ratio

Application	Dataset	Root Mean Squared Error*				Suppression Ratio(%)*			
		DBP	PLA	SAF	POR	DBP	PLA	SAF	POR
TUNNEL	Light	**18.867**	19.121	20.031	19.307	**99.74**	99.71	99.71	99.09
SOIL	Air Temperature	0.618	**0.613**	0.6196	0.794	**91.83**	91.77	91.79	89.30
	Soil Temperature	0.352	0.352	**0.3495**	0.361	98.80	98.82	**98.83**	97.83
INDOOR	Humidity	**4.494**	4.540	4.528	4.513	**99.50**	99.47	99.48	98.59
	Light	**23.980**	30.981	25.493	31.480	**97.58**	97.10	97.47	96.43
	Temperature	**1.972**	2.130	**1.972**	2.336	**99.60**	99.58	99.59	98.95
WATER	Chlorine	0.008	0.008	0.008	**0.007**	89.81	89.44	89.57	**92.58**

* **Underlined bold-face** numbers denote the lowest RMSE error or the highest suppression ratio.

by linear approximations as in DBP. Indeed, we chose this dataset as a sort of stress test for our technique. Although POR suppresses 2.77% more data reports than DBP, the performance of the latter is still very good, considering that: *i)* DBP still outperforms both PLA and SAF, *ii)* DBP's implementation is significantly easier and less memory-hungry than the other techniques, and therefore easier to integrate on resource-scarce WSN devices, and, *iii)* POR exhibits the worst performance in *all* other datasets, up to 2.53% less reports suppressed w.r.t. DBP in SOIL.

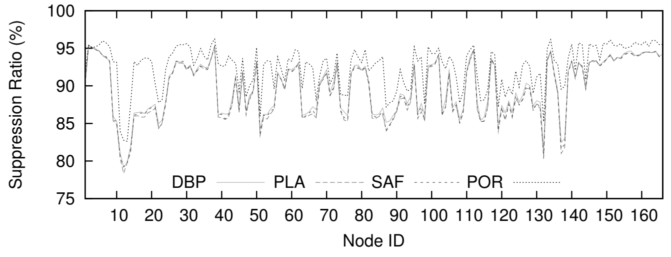
Table 2 shows the *aggregate* data suppression rate, but different nodes enjoy different SR values, based on the trends they observe in the sensed data. Fig. 4(a) provides a concrete view of this statement in WATER, our worst-case dataset, by showing SR for each node. Fig. 5 provides a more intuitive view on the same dataset by plotting the cumulative distribution function (CDF): a point on the curve represents the *number* of nodes, on the x-axis, whose SR is less than or equal to the one on the y-axis. The charts confirm the non-uniformity of data suppression, and show again that POR is consistently more efficient at suppressing reports than the other techniques, a consequence of the particular nature of the WATER dataset, as already mentioned. The lack of detailed information about the deployment of nodes and the trends of the physical phenomena observed, and the inability to run specific tests, prevents us from providing more in-depth observations in WATER, as well as SOIL and INDOOR.

On the contrary, in TUNNEL we do have all the information above. Fig. 4(b) shows the data suppression rate for the individual nodes in the tunnel. The chart is split in two to remind the reader that the deployment is constituted by two parallel lines of WSN devices, arranged as shown in Fig. 3. Indeed, the node placement motivates the difference in performance among the various nodes. The nodes in the tunnel interior are only marginally affected by the outside lighting conditions; the light data they sense is determined by the rather constant illumination provided by the tunnel lamps. All data prediction techniques are very effective in this case. On the other hand, the nodes near the entrance are subject to variations in light that can be also quite abrupt (e.g., upon sunrise) and that, contrary to the WATER dataset, are consistently predicted less effectively by POR.

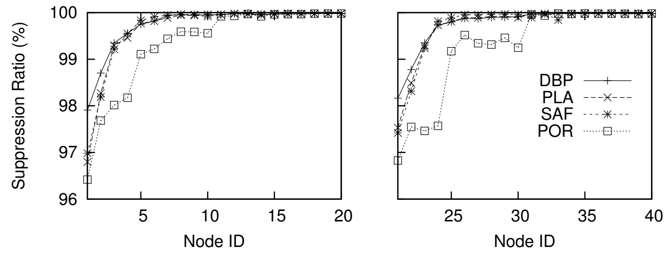
4.3 DBP in Action

In this section we take a closer look at the operation of DBP, showing that our technique can satisfy the error and delay tolerance requirements set by applications. We focus most of our discussion on the TUNNEL dataset, as it is the one for which we have most information, and occasionally compare with the WATER dataset, which is the worst case for DBP.

We begin by analyzing DBP in the small, dissecting the operation of a single node over a single day of operation for both these datasets, as shown in Fig. 6. In WATER, we chose node 1 because it has an average suppression ratio w.r.t. other nodes in the same deployment. In TUNNEL, we choose node 1 because, as shown in Fig. 3, it is placed at the tunnel entrance where, in comparison with nodes in the



(a) WATER



(b) TUNNEL

Fig. 4. DBP vs. state-of-the-art techniques on individual nodes in WATER and TUNNEL.

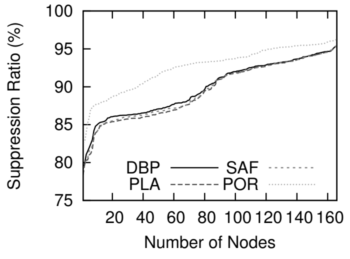
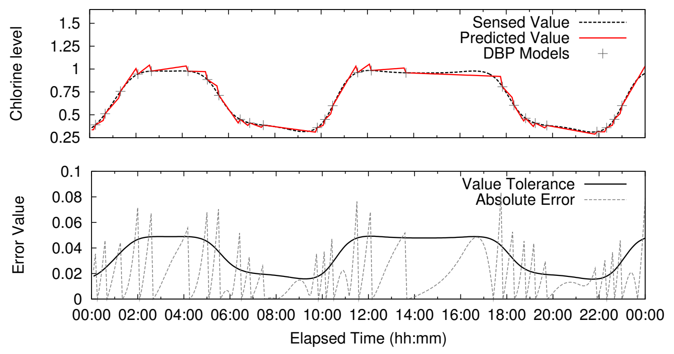
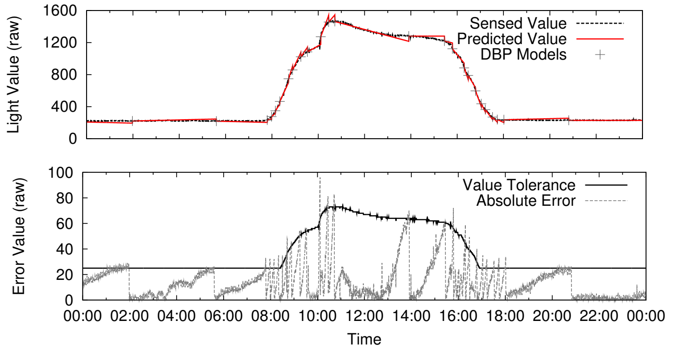


Fig. 5. A different view on Fig. 4(a): CDF of suppression ratio for the individual nodes in WATER.



(a) WATER.



(b) TUNNEL.

Fig. 6. Absolute values (top) and error (bottom), from the WATER and TUNNEL applications with DBP.

interior, most of the changes in light readings occur. The top charts in Fig. 6 show the values sensed by these nodes both in the original case where data is reported periodically (every 5′ for WATER and 30 s for TUNNEL, according to Table 1) and when DBP is applied. In the latter case, the cross points indicate the generation of a new model, while the lines between the points show the values automatically calculated at the sink from those models. The two datasets exhibit different trends: while the values in TUNNEL reflect the light changes induced by sunrise and sunset, the values in WATER are affected only by the concentration of chlorine, which is set arbitrarily by the simulation from which the dataset is extracted. Nevertheless, the charts show that DBP is able to predict very closely the actual values in both cases, while suppressing the majority of messages. For instance, in TUNNEL $2,880$ messages are sent without DBP, against only 25 messages with DBP: a suppression ratio of 99.13%.

As expected, most DBP models are generated in correspondence of slope changes in the value trends. Interestingly, in TUNNEL these are almost all concentrated around sunrise and sunset: the rest of the time, DBP generates very few models. These observations are confirmed on a global scale by Fig. 7, where we show the overall number of models generated by *all* the nodes in TUNNEL, over time. To measure this, we divide our 24-hour experiment into 5′ intervals and count the number of models generated by all nodes in each interval. The number of models in any 5′ interval reaches a peak of 10 after sunrise, a second peak of 4 around sunset, and remains well below this value during the rest of the day. At night, many intervals generate no models.

Finally, the bottom charts in Fig. 6 focus again on individual nodes as representative examples, analyzing the error of the values provided by DBP to the application. The solid line indicates the value tolerance set by our application requirements—$\varepsilon_V = (5, 0.0001)$ in WATER and $\varepsilon_V = (5, 25)$ in TUNNEL—while the lighter line shows the error of DBP as the difference between the predicted value and the sensed value. In most cases, the error falls below the value tolerance. Excursions above the value tolerance occur when data predicted at the sink, albeit incorrect, are within the time tolerance. In each of these cases, either subsequent values fell back below value tolerance or a new model was generated after the maximum number of incorrect reports ($\varepsilon_T = 2$ in our case) was exceeded. Interestingly, in many cases (e.g., at night in TUNNEL) one can see the absolute error growing for a while, then dropping and growing again. The drop in error corresponds to the generation of a new model, visible also in the top charts of Fig. 6. The growing error is because the DBP model is linear with a small, but non-zero slope, which is slightly off the measured light values that remain mostly constant. It is also worth noting that, in TUNNEL, the value tolerance at night is dominated by the absolute error ϵ^{abs}, while during the day it is dominated by the relative error ϵ^{rel}. Indeed, the light at the entrance of the tunnel at night amounts to only a few lux, while during the day it can easily exceed a thousand lux. This disparity motivates the use of two different value tolerances.

4.4 Impact of Parameter Settings

The previous evaluation shows that DBP performs well on our datasets, representative of real-world applications. However, we want to explore the parameter space for DBP, to understand the effect on the suppression ratio of changes to the error tolerances ε_V and ε_T, learning window m, and edge points l. Given the large number of combinations, we restrict ourselves to TUNNEL because, as discussed earlier, our direct knowledge of the application allows us to better interpret the impact of parameter changes.

Error Tolerance. Figs. 8(a)–8(c) show how SR changes at individual tunnel nodes, for various parameter combinations. Recall from Fig. 3 that nodes 1–20 are placed on the same North wall, while nodes 21–40 are on the South wall. We plot a line connecting the SR at each node, because this best highlights the trends as one proceeds from the entrance to the interior of the tunnel (e.g., from node 1 to 20 on the North wall).

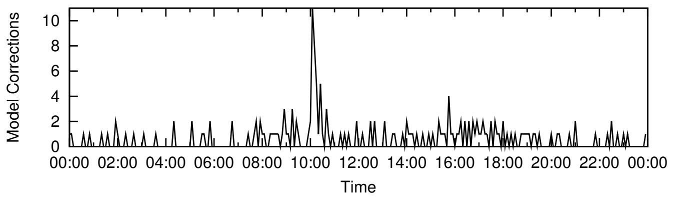


Fig. 7. Total number of DBP model updates in TUNNEL over 5-minute intervals, during a 24-hour experiment.

In Fig. 8(a) we vary the relative error ϵ^{rel} from 1% to 25%, keeping the absolute error constant $\epsilon^{abs} = 25$. By setting the time tolerance to $\varepsilon_T = 0$, we force all deviations from the value tolerances to be reported. To put these values in context, recall that the value tolerance ε_V is defined as the maximum between the relative and absolute errors, ϵ^{rel} and ϵ^{abs}. In Fig. 8(b) we fix $\epsilon^{rel} = 5\%$ and vary ϵ^{abs} between 0 and 50, keeping $\varepsilon_T = 0$. In Fig. 8(c), we use the value tolerance $\varepsilon_V = (5, 25)$ of our target application and vary ε_T between 0 and 4, i.e., from 0 to 2 minutes.

In all cases it is worth noting that, as expected, the biggest savings are seen at the nodes inside the tunnel, where light variations are rare, and absolute illuminance values are smaller. Under these conditions, the linear nature of DBP accurately models the linear nature of the data.

Interestingly, the trends for nodes 21–24 in Fig. 8(a) are due to the flickering of a lamp that introduced noise to the sensor readings. Nevertheless, DBP achieved $SR \geq 95\%$ even for these nodes. Further, in Fig. 8(b), we clearly see the need for both the absolute and relative value tolerances, as when the error tolerances are very low, e.g., $\epsilon^{abs} \in \{0, 10\}$, SR is off the bottom of the charts. This is because the light sensors themselves have an error that often takes them outside the small, fixed relative error $\epsilon^{rel} = 5\%$, triggering unpredictable model changes. Further, the flickering light introduces additional noise that DBP cannot compensate for with low error thresholds.

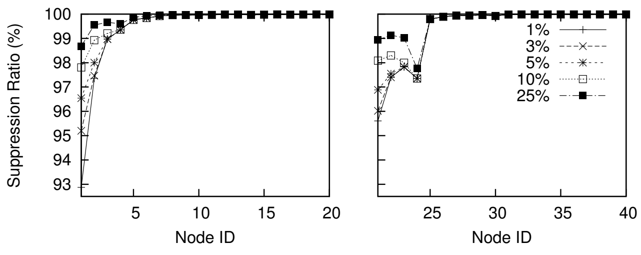
For each of these parameter combinations we also show, in Fig. 8(d), the average SR over all nodes. An increase in the value of ϵ^{rel} brings a near linear increase of SR. Instead, ϵ^{abs} and ε_T both achieve the greatest benefit at small values, with diminishing returns as the value increases. In the former case, the increase in SR progresses rapidly as ϵ^{abs} varies from 0 to 10, going from a suppression ratio of 88% to 98%; a further (and larger) ϵ^{abs} increase from 10 to 25 yields only an additional 2% increase of SR. Similarly, time tolerance reflects the fact that changes in light values are gradual, and thus introducing even a small delay $\varepsilon_T = 1$ achieves most of the possible gain.

In addition to the combinations in Fig. 8, we also computed the SR achieved with the *strictest* combination of the three parameters: $\epsilon^{rel} = 1\%$, $\epsilon^{abs} = 0$, and $\varepsilon_T = 0$. Even with these worst-case requirements DBP still suppresses, on average, 63% of the reports. More interesting is the *real* combination of parameters ($\epsilon^{rel} = 5\%$, $\epsilon^{abs} = 25$, and $\varepsilon_T = 2$) suggested by the TUNNEL engineers, and used in the rest of our experiments. In this case, the average suppression rate is a staggering 99.7%—SR is increased by almost two orders of magnitude w.r.t. reporting raw values.

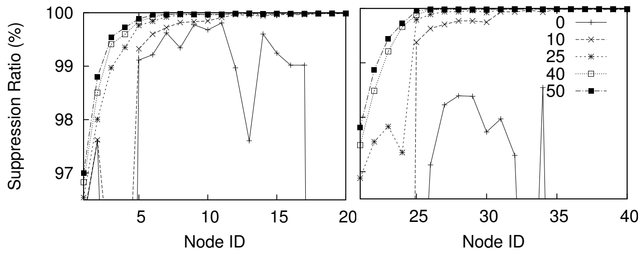
Learning Window and Edge Points. The learning window m and the edge points set l, used to compute the derivative, also have the potential to affect SR. Their value must be chosen based on the data properties and on the sampling interval. The intuition is that m should not be too big w.r.t. the sampling interval, to avoid ignoring important data trends. The number of edge points, $l \leq m/2$, should be small if the learning window contains few data points, although a value too small may increase the prediction error, especially in the presence of noise.

To verify the extent to which this intuition holds, we first applied DBP to our TUNNEL dataset with $m \in \{10, 15, 20, 25, 100\}$ and a fixed $l = 3$. In all cases, SR remains very high, between 99.7% and 99.8%. This is due to the nature of the TUNNEL dataset. The nodes at the entrance have large light variations during sunrise and sunset, when indeed larger m values yield a smaller SR. Interior nodes see minimal light variations throughout the day; a larger m yields a larger SR. The two effects cancel one another out, resulting in a constant performance, irrespective of m.

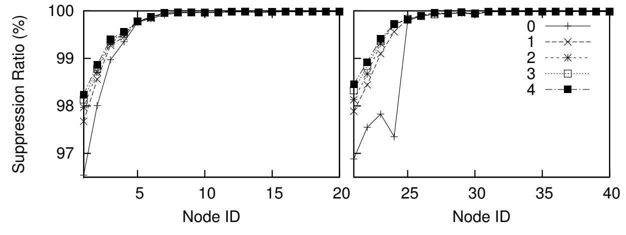
As this does not hold in general, we performed a similar analysis in the other datasets. Based on the sampling period (Table 1), we used the m values as in TUNNEL for the 3 INDOOR datasets, and $m \in \{6, 9, 12, 24, 48\}$ for the others. In all cases, we fixed $l = 3$. The impact, higher than in TUNNEL, is still negligible; in most datasets, the change in SR is $< 0.19\%$. WATER confirms to be the worst case; SR drops from 89.8% to 83.6%, showing that indeed too large of a learning window may miss important data trends, reducing the performance of DBP. Even in this case, however, the impact is limited; an 8-fold increase of m causes only a



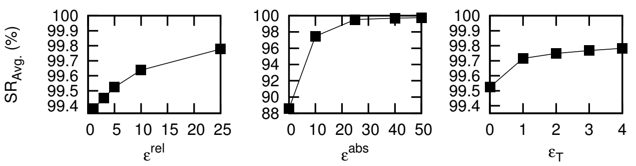
(a) Relative error ϵ^{rel} ($\epsilon^{abs} = 25$, $\varepsilon_T = 0$).



(b) Absolute error ϵ^{abs} ($\epsilon^{rel} = 5\%$, $\varepsilon_T = 0$).



(c) Time tolerance ε_T ($\epsilon^{rel} = 5\%$, $\epsilon^{abs} = 25$).



(d) Average suppression ratio for the combinations above.

Fig. 8. TUNNEL, in-field: Impact of error tolerance parameters on suppression ratio.

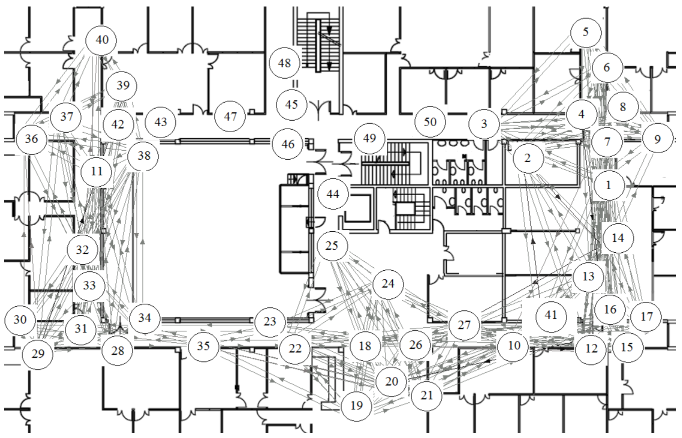


Fig. 9. Map of our 50-node testbed, including connectivity among the 40 nodes used for the TUNNEL dataset.

6.18% reduction in SR.

We similarly explored the impact of the number of edge points, by applying DBP with $l \in \{2, 3, 4, 5, 10\}$ and $m = 20$, to all datasets. Results show that the sensitivity of DBP to l is even lower. In TUNNEL, $SR = 99.7\%$ and differences are confined to the second decimal, due to the remarks above. In *all* cases, the change in SR is always $< 0.18\%$.

We conclude that, at least in our datasets, representative of mainstream WSN applications, DBP is only marginally sensitive to m and l, whose values must however be chosen relative to the sampling period.

5 A System-wide Evaluation

We now shift our focus from the application layer to the entire system, assessing the impact of data prediction with the full WSN network stack. As observed in the previous section, all the data prediction techniques studied achieve good results with all data sets, resulting in extremely low, aperiodic traffic. Therefore, here we focus our evaluation on two applications and datasets, TUNNEL and INDOOR temperature, that represent two extremes w.r.t. the traffic induced by DBP. In TUNNEL, model updates are concentrated at the nodes near the tunnel entrance, while in INDOOR, updates are more evenly distributed across all nodes. However, the reporting period is similar (30 s vs. 31 s; see Table 1) making a comparison meaningful.

We consider a system where DBP runs atop the mainstream WSN network stack composed of CTP [4], BoX-MAC [5], and TinyOS v2.1.1, a common choice in many WSN deployments, due its energy efficiency and relative ease of use. To evaluate the WSN behavior under different connectivity conditions, we perform experiments in two settings: an operational road tunnel, representing the real conditions of our target application, and an indoor testbed.

Tunnels are complex environments where factors such as road traffic affect network behavior. For example, we previously observed [15] that in the presence of high traffic, nodes consistently select parents on their same side of the tunnel, while at low traffic nodes across the tunnel are often selected. This is due to the interference caused by vehicles, which profoundly affects the shape and maintenance cost of the routing tree. For the in-field experiments reported here, we used the 40-node WSN shown in Fig. 3.

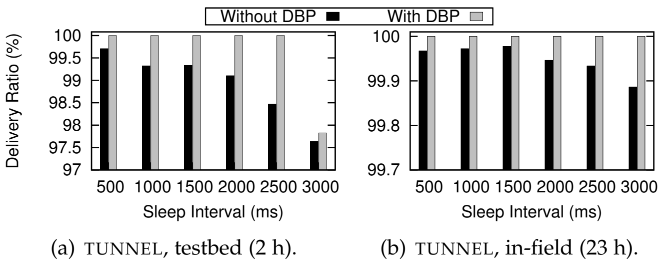
The testbed, instead, is composed of 50 TelosB nodes in a 60×40 m^2 office area, shown in Fig. 9. The INDOOR dataset uses all 50 nodes. TUNNEL uses only the first 40 that, at the -1 dBm power setting, yield a network topology forming three segments that approximate the linear tunnel topology, but with a larger diameter than our real tunnel.

To assess directly the impact of the network stack on the improvements theoretically attainable by DBP, we "replayed" the same data used in Section 4. As we could not re-execute the entire multi-day datasets with multiple parameter combinations, we selected a single 23-hour period from TUNNEL, ensuring variability in the vehicular traffic. Moreover, restrictions on the usage of the testbed forced us to run experiments only for a few hours. Therefore, when using TUNNEL data, we chose to focus on the sunrise period, the most challenging because values change dramatically. Finally, for INDOOR, we identified an interval for which the SR is close to the average one across the entire data set. Fig. 10 shows the number of models generated by each node in all cases. We begin the evaluation after DBP has been initialized, specifically after generation and transmission of the first model.

We next consider how application data delivery, network lifetime, and routing costs are affected by DBP, with the goal of understanding if improvements can be achieved by coordinating its functionality with the layers below it. All these metrics are deeply affected by the operation of the MAC layer, in particular the rate at which the radio duty cycles, which therefore becomes a key parameter in our experiments. At low sleep intervals, nodes frequently check the channel but find no activity, increasing idle listening costs. On the other hand, with long sleep intervals, the cost to transmit a packet increases. Specifically in BoX-MAC, transmission to a non-sink node takes on average half the sleep interval, due to the fact that the sender must transmit until the receiver wakes up, receives the packet, then acknowledges its reception [5]. Long transmission times also increase the probability of packet collisions among hidden terminals, further decreasing the delivery ratio and increasing energy consumption. The ideal sleep interval balances idle listening and active transmission costs. To identify the best interval for our applications, we ran our experiments with a range of values from 100 to 3000 ms.

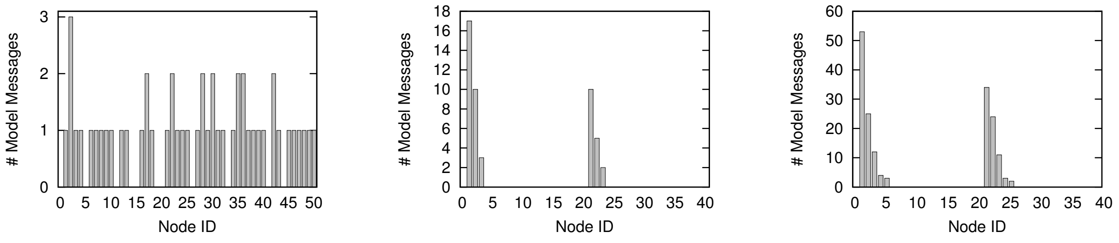
5.1 Data Delivery

DBP greatly reduces the amount of data in the network w.r.t. the baseline where all nodes send data periodically.



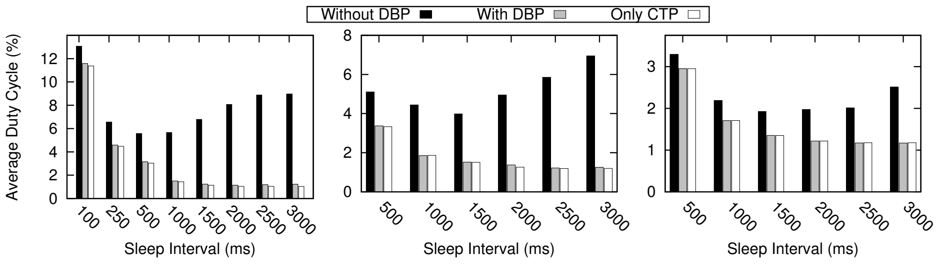
(a) TUNNEL, testbed (2 h). (b) TUNNEL, in-field (23 h).

Fig. 11. Delivery ratio for the TUNNEL dataset.



(a) INDOOR, testbed: SR=99.55% (2 h). (b) TUNNEL, testbed: SR=99.52% (2 h). (c) TUNNEL, in-field: SR=99.86% (23 h).

Fig. 10. Number of model update messages.



(a) INDOOR, testbed (2 h). (b) TUNNEL, testbed (2 h). (c) TUNNEL, in-field (23 h).

Fig. 12. Duty cycle. Note the varying y-axis scale.

The reduction in data transmitted reduces the probability of collisions, therefore increasing the delivery ratio. This is evident in Fig. 11, where the system with DBP loses fewer messages than without DBP. The figure shows only the results for TUNNEL; the testbed case is very similar to INDOOR. In all cases the delivery is very good, above 97%, but DBP actually always achieves 100%, except for the maximum sleep interval of 3000 ms in the testbed. In this case, a single model message was lost; however, as the absolute number of model changes is small, the total delivery ratio drops by almost 3%. Although this loss rate may be acceptable without DBP, losing a single DBP model has the potential to introduce large errors at the sink, as the latter will continue to predict sensor values with an out-of-date model until the next one is received. This suggests that, based on the target environment, dedicated mechanisms may be required to ensure reliability of model transmissions.

5.2 Lifetime

As the radio is one of the most power-hungry components, we use its duty cycle as a measure of system lifetime. Fig. 12 clearly shows that DBP enables significant savings at any sleep interval. Without DBP, the optimal sleep interval yielding the lowest duty cycle is 500 ms for INDOOR, and 1500 ms for both TUNNEL cases. The shorter optimal sleep interval of INDOOR is due to the extra data traffic from its additional 10 nodes; with more nodes, larger sleep intervals cause more collisions, resulting in a higher duty cycle.

In both cases, increasing the sleep interval beyond the optimal one decreases the idle listening cost, but correspondingly increases the transmission cost as the average transmission duration is half the sleep interval. This phenomenon instead bears a negligible effect in DBP where transmissions are greatly reduced. In this case, longer sleep intervals can be used to increase lifetime without affecting data delivery.

Focusing first on TUNNEL, Fig. 12(b) shows that in the testbed, with a sleep interval of 1500 ms (i.e., the best without DBP), DBP yields more than twice the lifetime of the no-DBP baseline—i.e., the WSN running DBP lasts twice as long, with the same MAC settings. Using the best sleep interval in both cases (i.e., 1500 and 3000 ms, respectively) yields a three-fold lifetime improvement. The energy savings for the in-field case, in Fig. 12(c), are less remarkable but still significant. The network diameter of the real tunnel is much smaller than the testbed due to the waveguide effect [15]; many direct, 1-hop links to the sink exist, leaving less room for improvement.

This impact of direct links to the sink is notable. As the sink is always on, it quickly receives and acknowledges packets, making transmissions from its children fast and low-energy. Fig. 13 shows this by separately plotting the (lower) duty cycle for nodes that were always 1-hop away from the sink. The plot shows only the no-DBP case, as with DBP *all* nodes reporting model changes were in direct range of the sink. Interestingly, by placing the gateway near the nodes with model changes, we further reduced communication costs. While this was not intentional in our case, it hints at a strategy for gateway placement to exploit

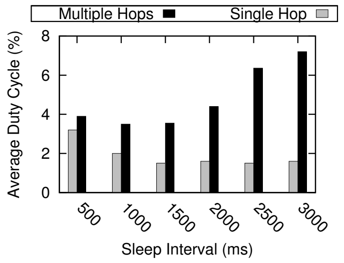


Fig. 13. TUNNEL, in-field: Duty cycle vs. sink distance, no DBP.

a priori knowledge of data patterns.

Finally, when applied to INDOOR, DBP achieves a five-fold lifetime improvement, larger than in TUNNEL. The reason can be grasped by noting that the duty cycle with DBP is ~ 1.2 in both datasets; DBP is effective in pushing both systems to their limit. Nevertheless, as noted earlier, INDOOR without DBP generates more traffic; this results in a smaller sleep interval for the optimal duty cycle, which is however higher than in TUNNEL (5.55 vs. 3.99). This difference without DBP, along with the similar duty cycle with DBP, determines the larger improvement in INDOOR.

5.3 Routing Costs

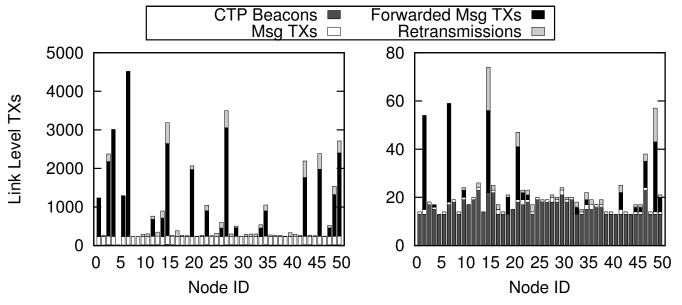
A natural question arises at this point: if DBP suppresses over 99% of the messages, why does the network lifetime increase "only" 3-fold for TUNNEL and 5-fold for INDOOR? This is due to the costs of the network stack: the idle listening and average transmission times of the MAC protocol, and the overhead of the routing protocol to build and maintain the data collection tree. As we already have evaluated the impact of the MAC, we now turn to routing.

To isolate the inherent costs (e.g., tree maintenance) of CTP, we ran experiments with no application traffic. The corresponding duty cycle is shown as *Only CTP* in Fig. 12; interestingly, the DBP cost is very close to the cost of CTP tree maintenance, regardless of the sleep interval. A finer-grained view is provided by Fig. 14, where we analyze the different components of traffic in the network, for both INDOOR testbed and TUNNEL in-field experiments, which have quite different topologies and durations. Without DBP, the dominating component is message transmission and forwarding; some nodes show several re-transmissions, while the component ascribed to CTP (i.e., the beacons probing for link quality) is negligible. When DBP is used, the number of CTP beacons remains basically unchanged. However, because application-level traffic is dramatically reduced, CTP beacons dominate the network traffic.

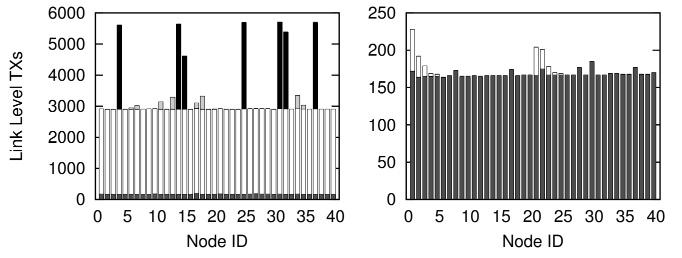
5.4 Cross-layer Routing Optimizations

The routing cost analysis above reveals that beacons, not application traffic, dominate the network traffic and therefore are the limiting factor for system lifetime. In CTP, the number of beacons, and therefore the cost of beaconing, is determined by an application of the Trickle algorithm [4], which sends one beacon at a random moment in a given time interval. This interval is initially small (0.125 s by default) to allow CTP to obtain and rapidly propagate accurate link cost estimates. However, to limit beaconing cost, if no link variations are detected the interval doubles, eventually reaching a large maximum value, 500 s by default. When beacons are triggered due to link cost variations, the interval shrinks back to the minimum, then gradually increases back to the maximum. In mainstream application environments, CTP spends most of the time at the maximum beacon interval. Here, we investigate how to reduce the beaconing cost, while still allowing CTP to function properly in the presence of link variations. To this end, we maintain the core Trickle mechanism but we experiment with larger maximum beacon intervals of 1000 s, 2000 s, and 4000 s. We hereafter refer to these values as 1x, 2x, 4x, and 8x.

The experiments we report are performed in our testbed

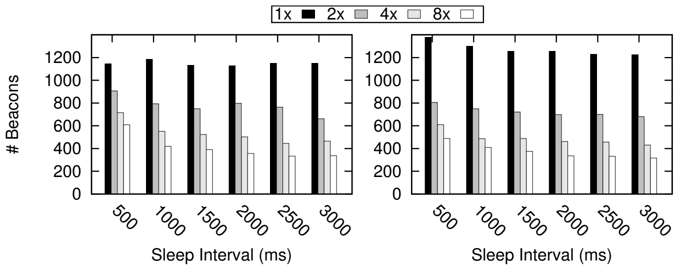


(a) INDOOR, testbed, without DBP. (b) INDOOR, testbed, with DBP.

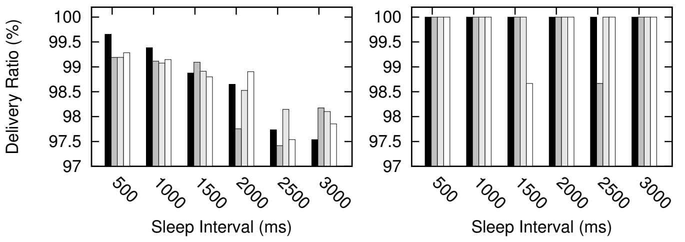


(c) TUNNEL, in-field, without DBP. (d) TUNNEL, in-field, with DBP.

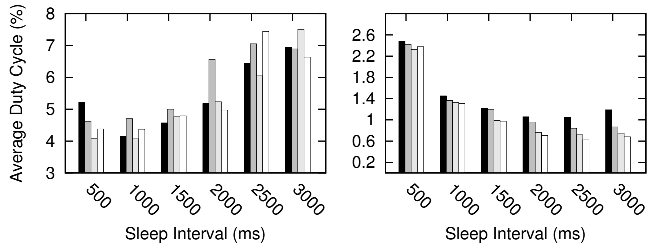
Fig. 14. Link-level transmissions with and without DBP for INDOOR, testbed (2 h) and TUNNEL, in-field (23 h). Note the varying y-axis scale.



(a) Without DBP. (b) With DBP.



(c) Without DBP. (d) With DBP.



(e) Without DBP. (f) With DBP.

Fig. 15. TUNNEL, testbed (4 h): beacon transmissions, delivery ratio, and duty cycle for different combinations of sleep intervals and beacon intervals. Note the different scale in the bottom charts for duty cycle.

for our main TUNNEL application, as results with the INDOOR data set are similar. Notably, these tests are longer than those reported earlier in this section, as CTP requires more time to reach a larger maximum beacon interval, specifically 2 hours at 8x. The 4-hour experiment duration is determined by restrictions on the testbed usage, and the need to keep the total experiment time manageable under the many combinations of parameters under consideration. Further, this set of experiments was also run at a later time w.r.t. those we presented earlier, and originally in [2]. Since then, the environment where the testbed is deployed underwent changes (e.g., a few walls were moved) that, albeit minor, affected connectivity.

The experiments we present here, therefore, are also the opportunity to validate our earlier results on a slightly different WSN setup and longer experiment duration. Fig. 15 shows our results, with different combinations of maximum beacon intervals and MAC-level sleep intervals. A comparison with Fig. 11–12 can be easily seen by focusing on 1x, the default maximum beacon interval. The trends are very similar to those observed earlier, with two major differences. First, the optimal sleep intervals without and with DBP are now 1000 ms and 2500 ms, respectively. Second, a comparison of the duty cycle for these optimal values shows a 4-fold improvement. The new setting is therefore more advantageous for DBP than in Section 5.2, where we obtained "only" a 3-fold improvement. Interestingly, our new setting is therefore a more challenging scenario for our goal of exploiting cross-layer routing optimizations to improve further over what DBP can achieve alone.

Looking at the rest of Fig. 15 we note that, as expected, the total number of beacons decreases as we increase the maximum beacon interval, yielding up to 70% fewer beacons at 8x. The number of beacons remains essentially the same with and without DBP, as shown in Fig. 15(a)–15(b). This is an expected consequence of this metric being tied to the stability of the network rather than the data traffic. Moreover, the impact of data delivery is dominated more by the MAC sleep interval than the maximum beacon interval, as shown in Fig. 15(c)–15(d). The trends are similar to those in Fig. 11(a), where the DBP system remains at 100% except for two cases, each with a single packet loss, while delivery without DBP degrades as the sleep interval increases.

On the other hand, increasing the maximum beacon interval bears a dramatic effect on lifetime. Without DBP, beacons are a small percentage of the overall network traffic. Therefore, as Fig. 15(e) shows, the effect of increasing the beacon interval does not bear a definite effect on duty cycle. Instead, with DBP this *always* provides a benefit. In particular, the duty cycle at the optimal sleep interval of 2500 ms is reduced by 40% when moving from 1x to 8x. If we compare this optimal DBP configuration (2500 ms, 8x) to the optimal configuration without DBP (1000 ms, 1x), the lifetime of the former is *seven times higher* than the latter. These results confirm that cross-layer tuning of the MAC-layer sleep interval and the routing layer beacon interval can lead to significant improvements.

Finally, we note that even with 4-hour experiments, the time for CTP to ramp up to the longest beacon interval is a significant fraction of the total experiment time, from 17 minutes at 1x to 135 minutes at 8x. Therefore, we expect the positive results shown here to be a conservative measure of the gains that can be attained in real deployments, where the effect of infrequent beacons dominate in the long term.

6 Related Work

The limited resources, variable connectivity, and spatio-temporal correlation among sensed values make efficiently collecting, processing and analyzing WSN data challenging. Early approaches use in-network aggregation to reduce the transmitted data, with later approaches addressing missing values, outliers, and intermittent connections [16]–[18].

Data prediction has been extensively studied, resulting in different techniques based on constant, linear [2], [9], [19], [20], non-linear [7] and correlation models [21]. Hung et al. [22] compare representative techniques from each category according to data reduction and prediction accuracy. The study concludes that constant and linear models outperform the others in the presence of small variations in the data. In line with these results, DBP uses a linear model. The other techniques based on probabilistic models [23], [24] approximate data with a user-specified confidence, but special data characteristics, such as periodic drifts, must be explicitly encoded by domain experts. In a similar parametric approximation technique [25], nodes collaborate to fit a global function to local measurements, but this requires an assumption about the number of estimators required to fit the data. In contrast, DBP requires neither expert domain knowledge nor lengthy training, but provides hard accuracy guarantees on the collected data. PAQ [20], SAF [7], and DKF [26] employ linear regression, autoregressive models, and Kalman filters respectively for modeling sensor measurements, with SAF performing best.

As an alternative to data modeling, some solutions seek to suppress reporting at the source by using spatio-temporal knowledge of data [27] or by identifying a set of representative nodes and restricting data collection to it [28]–[32]. Others take the remaining energy of individual nodes [33] into account. These approaches further reduce communication costs and can be applied in combination with DBP. Work on continuous queries for data streams studies the tradeoff between precision and performance when querying replicated, cached data [34]. Finally, several studies focus on summarizing streaming time series, showing that the choice of the summarization method does not greatly affect the accuracy of the summary [9]. In our experiments, we compared against PLA, as it can be efficiently computed.

Another technique that has recently become popular in WSN is compressive sensing (CS), e.g., [35], [36]. CS aims to *compress* a data sequence, *regardless* of the nature of the data, while DBP and existing data prediction techniques *suppress* data transmissions *based* on the nature of the data. However, the applications we target require hard error bounds (value and time tolerance). CS can only provide these guarantees for signals that are sparse in some space [37]; an a-priori analysis of the data traces is required to verify this precondition for applying CS, and can be difficult to achieve for streaming data. DBP has no such assumption; further, as shown in Section 4.4, even a non-optimal configuration of m and l bears little effect on efficiency, while always guaranteeing accuracy within the required error tolerance.

Finally, CS typically requires computation over a long data window (e.g., 200 samples) introducing a significant latency, problematic for several applications (e.g., including the control loop in our real-world TUNNEL application). Reducing the sample size is possible; this, however, increases packet transmissions and therefore reduces the attainable gains. DBP has no such limitations. Further, CS computation consists of multiplication/summation with a matrix (e.g., 10×10) whose storage induces a memory consumption higher than DBP.

The above data driven approaches have been evaluated theoretically, but no prior work explores the real effect of the network stack on energy savings. Network-level energy savings approaches can be classified into MAC level, cross-layer, or traffic-aware.

Low power MAC protocols [38] reduce overhead by limiting idle listening, overhearing, collisions and protocol overhead. Low-power listening protocols, e.g., BoX-MAC [5], dominate real deployments due to their availability, simplicity and effectiveness. However, as we have shown, the sleep interval must be carefully tuned.

Vertical solutions crossing network layers achieve extremely low duty cycles. Dozer [39] achieves permille (0.1%) duty cycle by taking a TDMA-like approach with scheduled transmissions. Unfortunately, Dozer does not scale well and is prone to choose poor quality parents. Koala [40] achieves similar low duty cycles, but by explicitly accepting delays between data generation and delivery. Koala is characterized by long periods of very low-power local data sampling followed by brief, high-consumption data collection intervals. While the energy savings are significant, the delays are not acceptable for closed-loop systems like our tunnel.

Other techniques [41], [42] adapt sleep schedules according to traffic statistics. Unfortunately, the data modeling approaches outlined above, of which DBP is an example, are difficult to predict due to the variability of the application data itself and the interaction with the modeling technique.

7 Conclusions

Data prediction exploits the fact that many applications can operate with approximate data, as long as it is ensured to be within certain limits of the actual data. This allows huge reductions in communication.

We applied our novel technique, DBP, to over 13 million data points from 4 real-world applications, showing that it suppresses up to 99% of the application data, a performance often better than other approaches despite that DBP is simpler and places minimal demands on resource-scarce WSN devices. The *practical* usefulness of DBP is reinforced by our system-wide evaluation, showing that with a properly tuned network stack, DBP can improve system lifetime seven-fold w.r.t. mainstream periodic reporting.

Our results suggest that further reductions in data traffic would have little impact on lifetime, as network costs are dominated by control operations. Therefore, improvements must directly address the extremely low data rates of DBP, e.g., by considering radically different network stacks. Further, data loss in prediction-based systems has the potential to significantly increase application errors. Therefore, reliable transport mechanisms must be revisited to ensure application-level quality.

Acknowledgments

This work was partially funded by the Autonomous Province of Trentino and by the European Institute of Innovation & Technology (EIT ICT Labs—Activity 12175 and 12149).

References

[1] T. Palpanas, "Real-time data analytics in sensor networks," in *Managing and Mining Sensor Data*, C. Aggarwal, Ed. Springer, 2012.

[2] U. Raza, A. Camerra, A. Murphy, T. Palpanas, and G. Picco, "What Does Model-driven Data Acquisition Really Achieve in Wireless Sensor Networks?" in *Proc. of the 10^{th} IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, 2012.

[3] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proc. of the Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2005.

[4] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "The collection tree protocol," in *Proc. of the Int. Conf. on Embedded Networked Sensor Systems (SenSys)*, 2009.

[5] D. Moss and P. Levis, "BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking," Stanford Information Networks Group, Tech. Rep. SING-08-00, 2008.

[6] M. Ceriotti, M. Corrà, L. D'Orazio, R. Doriguzzi, D. Facchin, S. Guna, G. P. Jesi, R. L. Cigno, L. Mottola, A. L. Murphy, M. Pescalli, G. P. Picco, D. Pregnolato, and C. Torghele, "Is there light at the ends of the tunnel? Wireless sensor networks for adaptive lighting in road tunnels," in *Proc. of the Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2011.

[7] D. Tulone and S. Madden, "An energy-efficient querying framework in sensor networks for detecting node similarities," in *Proc. of the Int. Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 2006.

[8] Life Under Your Feet Project, lifeunderyourfeet.org/en/src/.

[9] T. Palpanas, M. Vlachos, E. J. Keogh, and D. Gunopulos, "Streaming time series summarization using user-defined amnesic functions," *IEEE Trans. on Knowledge Data Engineering*, vol. 20, no. 7, 2008.

[10] EPANET, www.epa.gov/nrmrl/wswrd/dw/epanet.html.

[11] S. Papadimitriou, J. Sun, and C. Faloutsos, "Streaming pattern discovery in multiple time-series." in *VLDB*, 2005, pp. 697–708.

[12] J. Berry, W. E. Hart, and C. A. Phillips, "Sensor placement in municipal water networks," *J. Water*, vol. 131, 2003.

[13] S. Papadimitriou, F. Li, G. Kollios, and P. S. Yu, "Time series compressibility and privacy," in *VLDB*, 2007, pp. 459–470.

[14] J. Sun, S. Papadimitriou, and C. Faloutsos, "Online latent variable detection in sensor networks," in *ICDE*, 2005.

[15] L. Mottola, G. Picco, M. Ceriotti, S. Guna, and A. Murphy, "Not All Wireless Sensor Networks Are Created Equal: A Comparative Study On Tunnels." *ACM Trans. on Sensor Networks (TOSN)*, vol. 7, no. 2, 2010.

[16] L. Gruenwald, M. S. Sadik, R. Shukla, and H. Yang, "DEMS: a data mining based technique to handle missing data in mobile sensor network applications," in *Proc. of the Int. Conf. on Data Mgmt. for Sensor Networks (DMSN)*, 2010.

[17] A. Deligiannakis, Y. Kotidis, V. Vassalos, V. Stoumpos, and A. Delis, "Another outlier bites the dust: Computing meaningful aggregates in sensor networks," in *ICDE*, 2009.

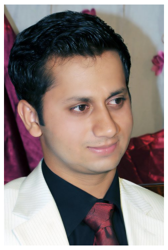
[18] W. Wu, H.-B. Lim, and K.-L. Tan, "Query-driven data collection and data forwarding in intermittently connected mobile sensor networks," in *Proc. of Int. Conf. on Data Mgmt. for Sensor Networks (DMSN)*, 2010.

[19] E. Gaura, J. Brusey, M. Allen, R. Wilkins, D. Goldsmith, and R. Rednic, "Edge mining the internet of things," *Sensors Journal, IEEE*, vol. 13, no. 10, pp. 3816–3825, Oct 2013.

[20] D. Tulone and S. Madden, "PAQ: Time series forecasting for approximate query answering in sensor networks," in *Proceedings of the European Wkshp. on Wireless Sensor Networks (EWSN)*, 2006.

[21] S. Gandhi, S. Nath, S. Suri, and J. Liu, "Gamps: Compressing multi sensor data by grouping and amplitude scaling," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '09. New York, NY, USA: ACM, 2009, pp. 771–784.

[22] N. Q. V. Hung, H. Jeung, and K. Aberer, “An evaluation of model-based approaches to sensor data compression,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 11, pp. 2434–2447, Nov 2013.
[23] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong, “Model-driven data acquisition in sensor networks,” in *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, 2004.
[24] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, “Approximate data collection in sensor networks using probabilistic models,” in *ICDE*, 2006.
[25] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, “Distributed regression: an efficient framework for modeling sensor network data,” in *Proc. of the Int. Symp. on Information Processing in Sensor Networks (IPSN)*, 2004.
[26] A. Jain, E. Y. Chang, and Y.-F. Wang, “Adaptive stream resource management using Kalman filters,” in *Proc. of the Int. Conf. on Management of Data (SIGMOD)*, 2004.
[27] A. Silberstein, G. Filpus, K. Munagala, and J. Yang, “Data-driven processing in sensor networks,” in *Proc. of the Conf. on Innovative Data Systems Research (CIDR)*, 2007.
[28] Y. Kotidis, “Snapshot queries: Towards data-centric sensor networks,” in *ICDE*, 2005.
[29] Z. Zhou, S. Das, and H. Gupta, “Connected k-coverage problem in sensor networks,” in *Proc. of the Int. Conf. on Computer Communications and Networks (IC3N)*, 2004.
[30] M. C. Vuran, O. B. Akan, and I. F. Akyildiz, “Spatio-temporal correlation: theory and applications for wireless sensor networks,” *Computer Networks*, vol. 45, no. 3, 2004.
[31] H. Jiang, S. Jin, and C. Wang, “Prediction or not? An energy-efficient framework for clustering-based data collection in wireless sensor networks,” *IEEE Trans. on Parallel Distributed Systems*, vol. 22, June 2011.
[32] M. Hassani, E. Mller, P. Spaus, A. Faqolli, T. Palpanas, and T. Seidl, “Self-organizing energy aware clustering of nodes in sensor networks using relevant attributes,” in *Proc. of the Int. Wkshp. on Knowledge Discovery from Sensor Data*, 2010.
[33] R. A. F. Mini, M. D. V. Machado, A. A. F. Loureiro, and B. Nath, “Prediction-based energy map for wireless sensor networks,” *Ad Hoc Networks*, vol. 3, 2005.
[34] C. Olston, J. Jiang, and J. Widom, “Adaptive filters for continuous queries over distributed data streams,” in *Proc. of the Int. Conf. on Management of Data (SIGMOD)*, 2003.
[35] C. Luo, F. Wu, J. Sun, and C. W. Chen, “Efficient measurement generation and pervasive sparsity for compressive data gathering,” *IEEE Trans. on Wireless Communications*, vol. 9, no. 12, pp. 3728–3738, 2010.
[36] C. Caione, D. Brunelli, and L. Benini, “Distributed compressive sampling for lifetime optimization in dense wireless sensor networks,” *IEEE Trans. on Industrial Informatics*, vol. 8, no. 1, pp. 30–40, 2012.
[37] E. J. Candès, J. K. Romberg, and T. Tao, “Stable signal recovery from incomplete and inaccurate measurements,” *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.
[38] J. Rousselot, A. El-Hoiydi, and J.-D. Decotignie, “Low power medium access control protocols for wireless sensor networks,” in *Proc. of the European Wireless Conf. (EW)*, June 2008.
[39] N. Burri, P. von Rickenbach, and R. Wattenhofer, “Dozer: ultra-low power data gathering in sensor networks,” in *Proc. of Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2007.
[40] R. Musaloiu-E., C.-J. M. Liang, and A. Terzis, “Koala: Ultra-low power data retrieval in wireless sensor networks,” in *Proc. of the Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2008.
[41] X. Ning and C. G. Cassandras, “Dynamic sleep time control in wireless sensor networks,” *ACM Trans. on Sensor Networks*, vol. 6, June 2010.
[42] C. J. Merlin and W. B. Heinzelman, “Duty cycle control for low power listening mac protocols,” in *Proc. of the Int. Conf. on Mobile Ad-hoc and Sensor Systems (MASS)*, 2008.



Usman Raza is a PhD student at University of Trento and Bruno Kessler Foundation, working on energy efficiency for WSNs and Cyber Physical Systems. He completed his MSc in 2008 with a National Management Foundation Gold Medal from Lahore University of Management Sciences, Pakistan. He received his BS from National University of Computer and Emerging Sciences, Pakistan in 2004.

Alessandro Camerra is a technical specialist at IBM System and Technology Group and a student at the Politecnico of Milan, Italy. Before this he worked as research assistant at the University of Trento, Italy and as visiting researcher at the University of California, Riverside. His publications cover the area of time series indexing and wireless sensor network technologies.

Amy L. Murphy is a research scientist at the Bruno Kessler Foundation-IRST, a research center in Trento, Italy. Prior to this, she worked as an assistant professor at the University of Lugano, Switzerland and the University of Rochester, NY, USA. Her research interests include the design, specification and implementation of middleware systems for wireless environments including wireless sensor networks.

Themis Palpanas is a professor of computer science at the Paris Descartes University, France. Before that he worked at the University of Trento and the IBM T.J. Watson Research Center. He is the author of 8 US patents, three of which are part of commercial products. He has received an IBM Shared University Research award, 3 Best Paper awards, and has been General Chair for VLDB 2013.

Gian Pietro Picco is a Professor and Head of the Department of Information Engineering and Computer Science (DISI) at the University of Trento, Italy. His research spans the fields of software engineering, middleware, and networking, and is oriented in particular towards wireless sensor networks, mobile computing, and large-scale distributed systems. He is an associate editor for ACM Trans. on Sensor Networks (TOSN) and IEEE Trans. on Software Engineering (TSE).