An Adaptive Strategy for Quality-Based Data Reduction in Wireless Sensor Networks

Silvia Santini
Institute for Pervasive Computing
ETH Zurich
8092 Zurich, Switzerland
santinis@inf.ethz.ch

Kay Römer
Institute for Pervasive Computing
ETH Zurich
8092 Zurich, Switzerland
roemer@inf.ethz.ch

***Abstract*— Wireless sensor networks allow fine-grained observations of real-world phenomena. However, providing constant measurement updates incurs high communication costs for each individual node, resulting in increased energy depletion in the network. *Data reduction strategies* aim at reducing the amount of data sent by each node, for example by predicting the measured values both at the source and the sink node, thus only requiring nodes to send the readings that deviate from the prediction. While effectively reducing power consumption, such techniques so far needed to rely on a-priori knowledge to correctly model the expected values. Our approach instead employs an algorithm that requires no prior modeling, allowing nodes to work independently and without using global model parameters. Using the Least-Mean-Square (LMS) adaptive algorithm on a publicly available, real-world (office environment) temperature data set, we have been able to achieve up to 92% communication reduction while maintaining a minimal accuracy of 0.5 degree Celsius.**

I. Introduction

Wireless sensor networks offer the possibility of continuously monitoring a variety of real-world phenomena in a distributed fashion. Such monitoring applications could allow a user to observe the evolution in both time and space of some measurable quantity. The sensor nodes, being deployed over a given region of interest, constitute an irregular spatial sampling grid for the observed phenomenon. At each time instant, the information residing in the network is thus a snapshot of the observed phenomenon. However, reporting these snapshots to an interested user represents a significant communication overhead and energy consumption.

To conserve network resources – energy, network bandwidth, CPU usage – a series of recent papers propose different approaches to reduce the amount of data that need to be delivered to the user [2], [5], [8], [15], [16], [18]. Since radio communication is known to be the dominant factor of energy consumption in wireless sensor networks, most of the proposed approaches focus on reducing communication among nodes while maintaining some form of cooperation. To get a concrete idea about the impact of radio communication on the global power budget of a sensor node, consider that on the *Telos Sky* mote – a well-known prototyping platforms for wireless sensor networks – the current consumption of powering radio transceiver is about twenty times higher than operating the micro-controller only and about three orders of magnitude higher than keeping the micro-controller in idle mode [21].

One common approach to reduce communication overhead is to select, among all data produced by the sensor network, a subset of sensor readings that is delivered to the user such that the original observation data can be reconstructed within some user-defined accuracy. In [18] and a series of later works, for example, the data maintained at the central server are guaranteed to be within a certain interval of the actual sensor readings, since nodes are required to report their readings to the server if the value falls outside this interval. To further reduce energy consumption, some recent works [2], [5], [8] propose to exploit spatio-temporal correlation among data to identify a subset of sensor readings from which the remaining measurements can be predicted within a given minimal accuracy. Readings which can be predicted from already delivered data do not need to be reported to the central server, thus reducing communication. Prediction can be performed in both time and space for example on the basis of some pre-defined model, whose parameters can be either learnt from historical data [2] or assigned by virtue of a-priori knowledge [8].

While these techniques have been proven to be effective for reducing power consumption, they not only suffer from performance losses when the model becomes outdated, but are also not well-suited to follow fine grained changes in sensor readings. To avoid a rapid deterioration in the predicted values, such approaches thus need their models to be periodically validated and correspondingly updated, implying again increased communication costs.

In this paper, we present an alternative data-reduction strategy that exploits the Least-Mean-Square (LMS) adaptive algorithm. The LMS is an adaptive algorithm with very low computational overhead and memory footprint that – despite its simplicity – provides excellent performance. More importantly, unlike previous work, our approach does not require a priori knowledge or modeling of the statistical properties of the observed signals. Hence, our scheme can be applied to a variety of real-world phenomena without restrictions. Moreover, the proposed algorithm does not require nodes to be assisted by a central entity for performing prediction, since no global model parameters need to be defined. Due to these characteristics, our approach can be easily integrated with a variety of existing data collection approaches – including schemes that support in-network data aggregation.

We show that our strategy can significantly reduce the number of readings that a sensor node is required to report to a sink node, while ensuring that the user can reconstruct the original observation data within a pre-specified minimal accuracy e_{max}.

We will also show that our LMS-based prediction scheme can be hierarchically extended to perform a *joint* prediction over a block of readings from neighboring nodes. Since a joint prediction can capture spatial correlation among neighboring sensors, a further reduction in communication can be achieved.

The remainder of the paper is organized as follows: in Section II, we provide a brief overview of related work. In Section III, we introduce some basic concepts of adaptive filter theory and provide a detailed description of our adaptive, quality-based strategy for data reduction in wireless sensor networks. After evaluating our approach in Section IV and pointing out some open issues and directions for further research in Section V, we will draw conclusions in Section VI.

II. Related Work

In the last years the problem of data reduction in sensor networks has received growing attention from the research community and a number of interesting approaches have been proposed. Since data are collected in both time and space, most of the proposed approaches perform data reduction in either the time [7], [8], [18] or space domain [6], [15]. Only a few approaches proposed feasible mechanisms for taking into account both the spatial and the temporal domain [2], [5].

Pioneering work by Goel and Imilienski [5] suggested to visualize a snapshot of the sensor readings in the network as an optical image. Given this visualization, the authors adapt the MPEG standard for video compression for use in sensor networks. The basic paradigm requires a base station to monitor the readings from the sensors and generate a prediction-model, which is valid over a given time interval. The model is then propagated to the sensor nodes, which send their readings only if they significantly differ from those predicted by the model. A similar, model-driven approach is proposed by Deshpande *et al.* in [2]. Also in this case a spatio-temporal prediction model is learnt from historical sensor data and is then used to estimate sensor readings in the current time period. Eventually, the model estimation can be refined by interrogating the sensor network for some specific current readings whose estimation uncertainty is high. Guestrin *et al.* [6] proposed to build a model of the data in the network using *kernel linear regression* and let the nodes transmit only significant changes in the model coefficients instead of raw data. In [15] several sensor subsets are identified, so that the readings from a single subset are enough to predict the readings of the remaining subsets. Using the subsets in a round robin fashion allows the solicitation of one subset at a time, with subsequent energy savings.

All the above described approaches have in common that they require the presence of a central entity which collects and process sensor readings from neighboring nodes in order to extract an adequate prediction model. The sensor nodes are thus not able to autonomously perform data reduction since they are reliant on a centrally defined model. A natural drawback of these techniques is also that models can become outdated and thus unreliable, with subsequent accuracy losses.

In [8], Jain *et al.* propose to approach the data reduction problem using classical linear filter theory. They introduce the *Dual Kalman Filter* (DKF) architecture as a general linear solution to this problem. In the DKF approach, each remote source involved in a specific sensing task runs an instance of a Kalman filter and performs linear prediction on smoothed sensor readings, sending updates to a central server only when the prediction error exceeds some given threshold. At the central server there are as many Kalman filters running as the number of remote sources. In this way, the server is able to mirror the behavior of the data sources and thus to reconstruct the phenomenon observed locally at each sensor node using either the received real data, when available, or the computed predictions. However, in order to use the Kalman filter for data streams prediction given a sequence of noisy observations, a model of the observed phenomenon must be provided to the filter. This implies that both server and nodes must feed the Kalman filter with the same model to be able to work coherently. Other approaches, such as the ones presented in [1], [14], are also based on the prior definition of prediction models.

III. Our Approach

Our approach aims at improving upon the work proposed in [8] by implementing a non-model based adaptive prediction scheme that does not require a-priori knowledge of the observed phenomenon. In the following we briefly summarize the basic mechanism of the dual prediction approach to data reduction, before detailing our proposed LMS-based prediction scheme.

A. Prediction-Based Monitoring

Consider a stream of sensor data $\{x[k]\}$ that has to be transmitted from a data source (i.e., some sensor node) to a data sink (i.e., another sensor node; possibly the gateway node). A minimal error budget (accuracy) e_{max} is given and known both by the source and the sink, such that the sink requires to know a value in $x[k] \pm e_{max}$ rather than the exact value $x[k]$.

Instead of transmitting the complete data stream $\{x[k]\}$ from source to sink, the goal of a data reduction strategy is to selectively transmit some elements of the data stream only, such that the sink is able to reproduce the whole data stream with the given accuracy. This can be achieved by the dual prediction scheme introducing identical predictive filters both in the source and in the sink. Such a filter can produce an estimate of the next element in the data stream, given some previous elements in the data stream. Only if the predicted value differs from the actual value by more than the error budget, the value is transmitted to the sink. Otherwise, the sink uses its local filter to generate the same prediction as the filter in the source – without need for communication.

Consider for example a node s_i having collected and reported to the sink s_j the readings $\underline{x} = \{x[1], x[2], \ldots, x[k-1]\}$. Assume both the node and the sink applying the same prediction algorithm to this data set and therewith computing an estimation $\hat{x}[k]$ of the upcoming reading $x[k]$. Since the node s_i holds the actual sensor reading $x[k]$, it is able to compute the estimation error $e[k] = \hat{x}[k] - x[k]$ and compare it to the user defined threshold e_{max}. If the threshold is exceeded, the node reports the reading to the sink node. Otherwise, if the threshold is *not* exceeded, the node simply discards the reading $x[k]$ and avoids reporting it to the sink node, which interprets the missing reporting as an implicit acknowledgement of the goodness of the prediction[1] and thus includes the value $\hat{x}[k]$ in its readings vector $\underline{x}$ in place of the real sensor value $x[k]$. Since the node discards the real reading $x[k]$ and also includes $\hat{x}[k]$ in its readings vector $\underline{x}$, the node and the sink share, at each instant k, the same knowledge about the observed phenomenon x. The basic principle of our approach is thus to let the same prediction algorithm run on the available data at both the node and the sink.

As we already mentioned in Section II, Jain *et al.* propose a practical implementation of this dual prediction scheme that makes use of Kalman filters [8]. Even if Kalman filtering has been proven to be a successful technique for data stream prediction in noisy environments, its practical application requires a-priori knowledge about the statistical properties of both the phenomenon being observed and of the measurement noise. To work properly, node and sink must thus agree on a pre-defined process model, which introduces an additional overhead and limits the application of the filtering scheme to a specific subclass of phenomena.

The classic adaptive filter theory [10] offers an elegant, alternative solution for performing predictions over data streams without requiring a-priori knowledge about the statistical properties of the phenomena of interest. Adaptive filters are indeed typically applied in environments where signals with unknown or non-stationary statistic are involved and appear for this reason particularly suited to be used in highly dynamic systems like sensor networks. Among the variety of existing adaptive algorithms we choose to use the Least-Mean-Square (LMS) algorithm [11] to implement the dual prediction scheme described above.

In Section III-C we will provide a brief introduction to the adaptive filter theory and the LMS-algorithm. Before going into further details, we would like to point out that this basic concept can be applied to a variety of different data collection approaches in sensor networks as described in the following Section III-B.

[1]Since wireless transmission in sensor networks is known to be unreliable, it is likely to happen that some of the node's reported readings never reach the sink node, thus causing a misalignment between the readings vector $\underline{x}$ at the node and the correspondent vector at the sink. For simplicity, we assume for the remainder of this section a loss-free communication link between the nodes s_i and the sink s_j. See section V for a discussion of some mechanisms that allow to relax this assumption.

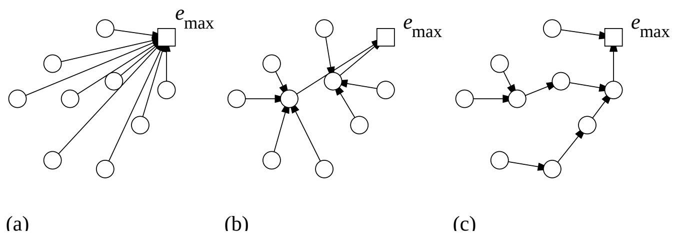


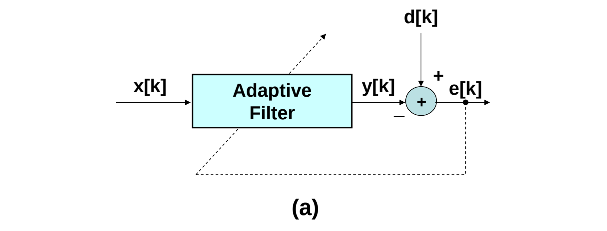
Fig. 1. Network model: (a) star network, (b) clustered network, (c) tree network.

B. Network Model

In the sensor networks literature, numerous different approaches to data collection have been explored. Figure 1 shows three typical network topologies used for delivering sensor data from sensor nodes (circles) to the gateway node (square). The conceptually simplest solution is a star topology depicted in part (a), where each sensor node delivers a stream of sensor data directly to the sink – possibly across multiple hops. Numerous approaches (e.g., LEACH [12]) employ a cluster structure depicted in part (b), where sensor nodes report data to a cluster head, the cluster head processes these data and delivers the result to the gateway – again, potentially accross multiple hops. Another common topology is the tree structure (e.g., TAG [16], Directed Diffusion [13]) depicted in part (c), where a sensor node receives data from its children and aggregates them with its own data, sending results on to its parent.

Our data reduction scheme can be employed in all of these scenarios. For this, we consider each pair of nodes that are directly connected by an arrow in Figure 1 as a pair of source and sink and introduce filters as described in the previous section. Hence, the data reduction is not only applied to the raw sensor data generated by a sensor node, but also to streams of aggregated data such as the aggregated data sent by a cluster head to the gateway in Figure 1(b). The latter is possible since no a-priori knowledge about the statistical properties of the aggregated data stream is needed by the filters. Also, the low overhead of the LMS filter enables a sensor node to run multiple instances of the filter (i.e., one for each of its neighbors in the network topology).

An interesting issue here is what error budget should be used by each of the source/sink pairs, given that the gateway is requested to produce data with an error budget e_{max}. To examine this issue, let us assume that each non-leaf node (i.e., the gateway in part (a) of Figure 1, clusters heads and the gateway in part (b), inner nodes and gateway in part (c)) implement an aggregation function $x[k] = f(x_1[k], x_2[k], \ldots)$, where $x_i[k]$ are the values obtained from neighbor node i at instant k. Assuming that the node implementing f is given an error budget e_{max}, error budgets e_i must be computed for each neighbor. If f is a simple selection function (e.g., $\min(x_i), \max(x_i)$, or $\{x_i\}$), then $e_i := e_{max}$ assuming that all neighbors produce identical data. However, f can also be a more complex function. For example, it might compute



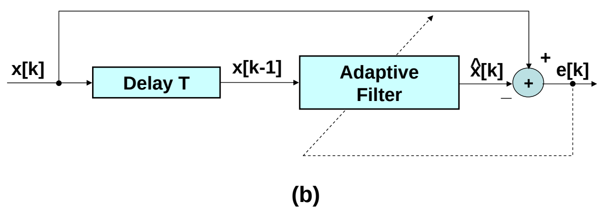


Fig. 2. Adaptive filter: (a) generic scheme, (b) as a prediction filter.

TABLE I
LMS-ALGORITHM

$y[k] = \underline{w}^t[k]\underline{x}[k]$	Filter output
$e[k] = d[k] - y[k]$	Error signal
$\underline{w}[k+1] = \underline{w}[k] + \mu\underline{x}[k]e[k]$	Weights adaptation

the location of a tracked object given magnetometer readings x_i of its neighbors. In this case, e_{max} will be a location accuracy specified in, e.g., meters, which must be translated into accuracy e_i of magnetometer data from neighbor i. The derivation of e_i from e_{max} is inherently dependent on f.

Computing the error budgets for a given network topology can typically be performed during query dissemination, where the query is delivered from the gateway to the sensor nodes along the reverse direction of the arrows in Figure 1. When a node receives a query containing the requested error budget e_{max}, it computes the e_i according to f and includes the respective value in the query message that is forwarded to the respective neighbor.

C. Adaptive Filters and the LMS Algorithm

Adaptive filters are typically used in applications in which signals with unknown or non-stationary statistics are involved. The generic structure of an adaptive filter is shown in Figure 2a. Basically, a linear adaptive filter takes at each step k a sample $x[k]$ of the input signal x and computes the filter output $y[k]$ as:

$$y[k] = \sum_{i=0}^{N-1} w_{i+1}[k] * x[k-i] \quad (1)$$

thus, as a linear combination of the last N samples of the input signal x, each one of them being weighted by the accordant filter coefficient $w_i[k]$. The output signal $y[k]$ is then compared to a reference signal $d[k]$ (d for *desired*), which is the signal the filter tries to adapt to. The error signal $e[k]$ is thus simply computed as the difference $e[k] = y[k] - d[k]$ and given as input to the adaptation algorithm, which accordingly updates the filter weights. These weights are modified at each time step k in order to satisfy a given optimality criterion which is typically the minimization of the Mean-Square-Error (MSE), i.e., the average error power $E\{e^2[k]\}$. Without going into details, we want to point out that the choice of the MSE as optimality criterion implies that the error function $J(\underline{w})$, which describes the dependency of the MSE from the filter weights $\underline{w}$, is a quadratic function, which has in most cases a unique absolut minimum point $\underline{w}^{opt}$, i.e., a unique optimal solution which minimizes the MSE. The filter weights are updated at each step k with the aim to iteratively approach this minimum point. The error signal $e[k]$ gives the adaptation algorithm a measure of the extent of the correction that needs to be applied to the filter weights in order to reduce, at the subsequent step $k+1$, the error power $E\{e[k+1]\}$.

If the statistics of the involved signals[2] were stationary and known a-priori, the set of optimal filter weights $\underline{w}^{opt}$ which minimizes the MSE could be computed trough the well-known Wiener-Hopf equation [20], [10]. Adaptive filters however, do not require a-priori knowledge on the signals statistics, rather they learn these statistics from the data and adapt to their changes by updating the filter weights $\underline{w}$. In this sense, adaptive filters provide a *tracking* capability, since they are able, in a non-stationary environment, to track time variations in the statistics of the input data, provided that these variations are sufficiently slow.

There is a huge number of adaptive algorithms which have been developed in the literature [10]. The choice of one algorithm over another is determined by the trade-off among different factors, like rate of convergence, robustness, computational complexity, numerical properties, among others. One of the most successfully applied adaptive algorithm is the so-called Least-Mean-Square algorithm (LMS). Despite of its simplicity, this algorithm provides very good performances in a wide spectrum of applications [11]. The LMS algorithm is basically defined through the three equations reported in Table I, where $\underline{w}[k]$ and $\underline{x}[k]$ denote the $N \times 1$ column vectors:

$$\underline{w}[k] = [w_1[k], w_2k, \ldots, w_N[k]]^T \quad (2)$$

$$\underline{x}[k] = [x[k-1], x[k-2], \ldots, x[k-N]]^T \quad (3)$$

The LMS algorithm, like any other adaptive algorithm, can be used to perform prediction when the general filter structure in Figure 2(a) is refined in the predictive structure of 2(b). The basic principle consists in delaying the current input value $x[k]$ by one step and use it as the reference signal $d[k]$. The filter computes an estimation $\hat{x}[k]$ of the input signal at the step k, as a linear combination of the N previous readings $\{x[k-1], x[k-2], \ldots, x[k-N]\}$:

[2]In particular, the autocorrelation matrix of the input signal and the cross-correlation vector of the input and the reference signal.

$$\hat{x}[k] = \sum_{i=1}^{N} w_i[k] * x[k-i]. \quad (4)$$

The prediction error is then computed and fed back to adapt the filter weights. For the adaptation process, two parameters need to be defined: the step-size μ, that appears in the weight update equation (reported in Table I) and the filter length N.

The step-size μ is a critical parameter since it tunes the convergence speed of the algorithm. There exists a practical criterium for a straightforward computation of this parameter from a small set of observations, as we will point out in the next section.

The number of filter weights, normally referred to as the *filter length N*, is also an important parameter for the computational load and memory footprint of the filter. From the equations reported in table I and recalling that $\underline{w}$ and $\underline{x}$ are $N \times 1$ vectors, it is straightforward to see that the LMS algorithm requires $2N + 1$ multiplications and $2N$ additions per iteration. In order to keep the computational load of the filter low, the number of weights N must be kept as low as possible. We will show in Section IV that the filter performs very efficiently even with very small filter lengths ($N = 4, \ldots, 10$). It is also important to notice that increasing the filter length does not necessarily improve the performance of the filter. In particular, increasing N above a theoretically determinable threshold value N^{opt} will result in a performance loss.

For a further detailed explanation of the characteristics of the LMS algorithm, we refer to [11]. In the following, we will explain how this algorithm allows the implementation of an efficient data reduction strategy.

D. Quality-Based Data Reduction Using the LMS Algorithm

Using the LMS algorithm for implementing the dual prediction scheme described in Section III-A allows to significantly reduce the amount of data a node is required to report to its sink in order to guarantee the user-defined error budget e_{max}. This reduction is achieved by letting the node switch as frequently as possible from its *normal* operational mode to a so-called *stand-alone* mode, in which the node does not need to report sensor readings to the sink. In order to be able to run the prediction algorithm, the node needs to go through an *initialization* phase. These three basic states of node operation are described in the remainder of this section.

1) Initialization mode: when a node receives a user query, it starts collecting and reporting data to the sink node, without performing prediction, as explained in Section III-A. In this phase, the node and the sink compute an estimation of the step-size μ. To ensure convergency, the step-size μ must satisfy the following condition [10]:

$$0 \leq \mu \leq \frac{1}{E_x} \quad (5)$$

where E_x indicates the mean input power computed as:

$$E_x = \frac{1}{M} \cdot \sum_{k=1}^{M} |x[k]|^2 \quad (6)$$

and M is the number of iterations used to train the filter [10]. Since the input mean power E_x is time-varying, an approximation $\bar{E}_x$ can be computed over the first N samples and used to compute the upper bound in inequality 5 above. In practical applications, choosing the step-size μ two orders of magnitude smaller than this bound, typically guarantees the robustness of the filter [17]. The filter length N is set to very small values that have proven to be efficient for the analyzed data sets, as we will show in Section IV. We reserve for future work the development of an on-line estimation technique of the optimal filter length N.

Once the initialization phase is completed, both the node and the sink will start performing prediction on the collected readings and operate in either *normal* or *stand-alone* mode, as explained below.

2) Normal mode: when working in normal mode, both the node and the sink use the last N readings to compute a prediction for the upcoming measurement, and update the set of filter coefficients $\underline{w}$ on the basis of the actual prediction error, using the equation given in Table I. Please note that the default start value for the filter coefficients is assumed to be $\underline{w}[0] = \underline{0}$. This assumption is particularly relevant, since unlike other adaptive algorithm, the LMS approach ensures that multiple instances of the filter fed with the same sequence of data and sharing the same set of initial weights $\underline{w}[0]$ (and, of course, the same values for N and μ), will compute the same set of filter coefficients and thus, the same predictions, at each time instant k.

As long as the prediction error exceeds the user defined error budget e_{max}, the node keeps working in normal mode, thus collecting and reporting its readings to the sink. When the prediction error drops below the threshold e_{max} for N consecutive steps, the node will switch to *stand-alone* mode. As long as the node remains in the *normal* mode, the sink will let the prediction filter run over the received sensor readings, in order to update the filter weights $\underline{w}$ coherently with the node.

3) Stand-alone mode: when working in *stand-alone* mode, the node keeps collecting data and computes the prediction at each time step. As long as the prediction error remains below the given threshold e_{max}, the node discards the reading and feeds the filter with the prediction $\hat{x}[k]$ instead of with the real data $x[k]$. This will ensure the state of the filter at node side to remain consistent with the state of the filter at the sink side. Please note that feeding the filter with its own prediction causes the prediction error e to be zero and thus the filter weights to be left unchanged. This is another advantage of using this technique since staying in *stand-alone* mode, the

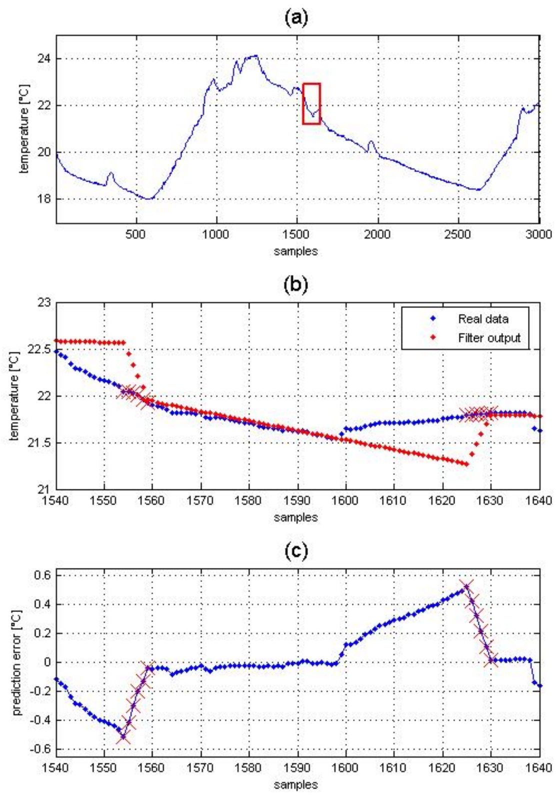


Fig. 3. LMS prediction : (a) real sensor readings, (b) real and predicted sensor readings, (c) prediction error.

node can omit updating the weights, thus saving half of the computational overhead.

If at time instant k the node observes the prediction error exceeding the threshold e_{max}, it will report the reading $x[k]$ to the sink and switch back to *normal* mode.

While the node operates in *stand-alone* mode, the sink, receiving no readings from the node[3], implicitly assumes the predicted readings being a good enough approximation of the real readings and keeps running the prediction filter on these values.

Figure 3 illustrates how our scheme works. We let our algorithm (with $N = 5$ and $\mu = 10^{-5}$) run on a set of temperature readings obtained from a real world sensor [22], as shown in Figure 3(a). Figure 3(b) shows a detailed view of the outlined area in subfigure (a), with an overlapping plot of the correspondent filter output. Subfigure (c) shows the prediction error of the datapoints in subfigure (b), including highlights (with a cross) for those readings which the node effectively needs to report to the sink in order to guarantee an accuracy e_{max} of $\pm 0.5°C$. We see that as soon as the error exceeds the given threshold, the corresponding sensor reading is sent

[3]For this, the sink has to decide whether a message has been sent by the source or not. We assume the source to collect and (eventually) send readings at regular time intervals, such that the sink can easily recognize the absence of a message.

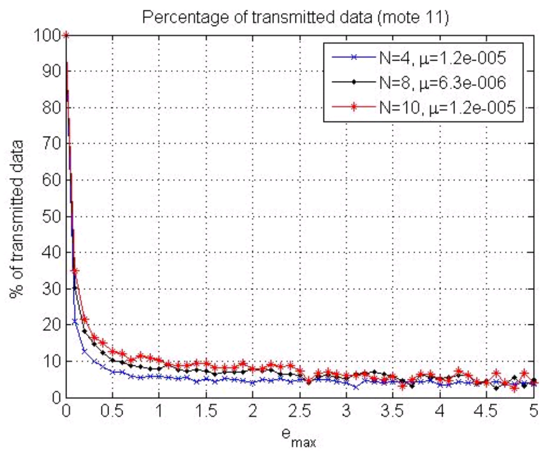


Fig. 4. Percentage of data transmitted by mote 11.

to the sink and the filter restarts adapting to the real data, thus causing the prediction error to decrease. As soon as the error remains below $\pm 0.5°C$ for at least $N = 5$ readings, the node stops reporting data (i.e., switches again to *stand-alone* mode).

Note that an *outlier detection* mechanism could be easily embedded into our scheme. Since outliers are likely to appear [4] and their presence could disturb the operation of the prediction filter, it is good practice to include some automation in the algorithms for their detection. In our case, an adequate threshold may be defined either by the user or by the node itself (e.g., as a multiple of the mean error). A reading whose correspondent prediction error is larger than this threshold will be classified as an outlier and discarded. In this case, the discarded data could be replaced by the corresponding prediction.

IV. Evaluation

In this section we present the results obtained when applying our LMS dual prediction scheme to real world data. Some challenges related to the practical implementation of the proposed algorithm, as well as some open issues which are currently object of further research will be pointed out in the next section.

We tested our LMS-based data reduction strategy on a set of real world data publicly available at [22]. Once every 31 seconds, humidity, temperature, light and voltage values where collected from 54 Mica2Dot sensor nodes [24] deployed in the Intel Berkeley Research lab [23] between February 28 and April 5, 2004. To report our results, we picked 4 of these 54 motes, namely motes 1, 11, 13, and 49 which were distributed in different sectors of the deployment area. We applied our scheme to the data reported by the temperature sensors of these four motes between March 6 and 9.

In Figure 4 we report the percentage of sensor readings that mote 11 would need to report as the error budget

[4]Consider for example that, in the case of normally distributed data, about 1 in 150 observations will be a mild outlier and only about 1 in 425,000 an extreme outlier.

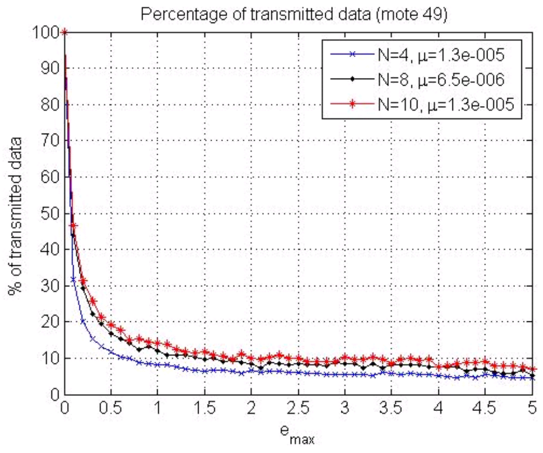


Fig. 5. Percentage of data transmitted by mote 49.

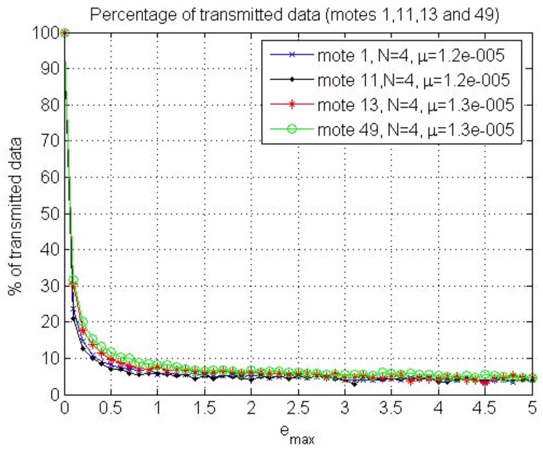


Fig. 6. Percentage of transmitted data by motes 1, 11, 13, and 49, for a filter length $N = 4$.

e_{max} increases, for different filter lengths N. As it can be seen, a minimal accuracy of $0.5°C$ can be guaranteed while transmitting only about 10% of the collected sensor readings. This significant data reduction is due to the optimal tracking capability of the LMS algorithm. Moreover, no significant changes in the performances are observed when varying the number of filter weights from $N = 4$ to $N = 10$. Since the number of operations to be performed at each time step grows proportionally[5] with N, this value should be kept as small as possible. The tested values of N allow to keep extremely low the computational overhead and memory footprint of the algorithm. With $N = 4$, for example, the node must perform at most 17 operations each 31 seconds and needs to store in addition to the 4 filter coefficients and the filter parameters, only the last 4 sensor readings. Analogous results have been obtained testing our algorithm on mote 49, as reported in Figure 5. However, since the data reported by mote 49 are more spiky than those of mote 11, a small deterioration in the performances can be seen. Finally, Figure 6 shows the performances obtained with two additional data sets, namely those collected by mote 1 and 13. Mote 1 is located far away from both mote 11 and 49, while mote 13 lies in the same room as mote 11. Also with these data sets we obtained very good results in terms of data reduction.

[5]Recall from section III-C that the computational cost per iteration of the LMS algorithm is $4N + 1$ when the node operates in *normal* mode and $2N$ in *stand-alone* mode.

V. Open issues and future work

While we could show that our LMS-based data reduction strategy provides good prediction performance with no a priory assumptions and low overhead, there are also some potential problems and issues that need to be addressed. These will be discussed below.

1) Communication model: Earlier in this paper we assumed loss-free communication links between sink and source to implement the dual prediction scheme. However, this is not a realistic assumption in real-world deployments [22]. Hence, we need to provide appropriate mechanisms to make our scheme robust to message loss.

Let us consider the impact of message loss on our prediction scheme. Firstly, if a message is lost, the sink will use its filter to predict a value which is not within the error budget e_{max}. Secondly, the filters in source and sink will get out of synchronization, that is, they will output different predictions. Even when the sink receives a message eventually, the sink's filter will produce wrong predictions. Hence, it is important to provide a mechanism whereby the sink can detect lost messages and resynchronize its filter with the source. This can be achieved, for example, by including a sequence number in each message. If the sink detects a jump in the sequence number, message loss is assumed and the source is forced into *initialization* mode. However, this approach does not represent a satisfactory solution due to the incurred communication overhead.

We are currently examining an alternative approach, where, based on the actual packet loss ratio p of the link between source and sink (using either an a priori estimate or in situ measurements), a modified *equivalent* error budget $e^{eq}_{max} = f(p, e_{max}) \leq e_{max}$ is computed and used instead of e_{max}, such that, on average, accuracy e_{max} can be achieved even in case of message loss. This solution will of course cause some deterioration with respect to the performances presented in Section IV, but will also allow to apply and test our approach in a more realistic framework.

2) Node failures: In typical sensor network deployments, sensor nodes are subject to frequent crashes, battery depletion, and other temporary or permanent failures. If the source node fails due to these reasons, the sink would continue to output predictions. To limit the impact of such failures, the sink needs a mechanism to detect source failure. This can be easily achieved by forcing the source node transmit at least one sensor reading every K instants, even though this would not be necessary otherwise. If after K instants from the last transmission the sink does not receive this message, source failure is assumed.

3) Spatial prediction: In this paper, we presented the potential of the LMS algorithm in performing the prediction of a single sensor reading based on N previously observed readings. In other words, we focused on prediction of a single node in the time domain. However, since the LMS filter can be extended to work on blocks of data instead on single values, we envision an interesting spatial extension of our prediction scheme. Consider for example a cluster head s_i receiving sensor readings from N neighboring sensor nodes and assume these readings are required to be reported to the gateway (cf. Figure 1 (b)). At each time instant k, node s_i holds a block of N values representing either the actual or the predicted readings of the N nodes. This block of readings can be used to train a multiple-error LMS adaptive filter [3], which will jointly optimize the prediction error over the block of given input data. Since this kind of joint filtering will capture spatial correlation among readings, a further reduction in communication overhead can be achieved, since the node s_i will report its readings to the sink only if the joint prediction error will exceed the given threshold. The concrete implementation of this joint filtering scheme offers interesting perspectives for future work.

VI. Conclusions

We considered wireless sensor network applications that require a continuous delivery of sensor readings at regular time intervals kT. We presented an adaptive approach that allows to significantly reduce the amount of data that needs to be transmitted, while ensuring that the original observation data can be reconstructed within a pre-specified accuracy e_{max}. Our approach is based on an efficient prediction technique using the LMS adaptive algorithm at both the source and sink of a data stream. Through tests on real-world data we demonstrated the effectiveness of this approach.

Unlike previous work, our approach is lightweight and does not assume a-priori knowledge about statistical properties of the observed phenomena and thus lends itself well to many practical applications. Also, the approach can be easily integrated with a variety of different data collection approaches including clustering techniques and in-network data aggregation.

Future work includes an extension to spatial prediction by means of a block variant of the LMS filter and an improved handling of message loss.

VII. Acknowledgments

We gratefully acknowledge Friedemann Mattern and Marc Langheinrich for valuable comments and suggestions on draft versions of this paper.

The work presented in this paper was partially supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

[1] R.Cheng, D.V. Kalashnikov and S.Prabhakar: Evaluating Probabilistic Queries over Imprecise Data. In Proceedings of the ACM SIGMOD/PODS Conference (SIGMOD '03), San Diego, USA, June 2003.

[2] A. Deshpande, C. Guestrin, S.R. Madden, J.M. Hellerstein and W. Hong: *Model-Driven Data Acquisitionin Sensor Networks.* In Proceedings of the 30th VLDB Conference (VLDB '04), Toronto, Canada, 2004.

[3] S.C. Douglas: *Analysis of the Multiple-Error and Block Least-Mean-Square Adaptive Algorithms.* IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 42, No. 2, February 1995.

[4] D.Estrin, L. Girod, G. Pottie and M. Srivastava: *Instrumenting the World With Wireless Sensor Networks.* International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01), Salt Lake City, Utah, May 2001.

[5] S.Goel and T. Imielinski. *Prediction-based monitoring in sensor networks: Taking lessons from mpeg.* In ACM Computer Communication Review, 31(5), 2001.

[6] C. Guestrin, P. Bodik, R. Thibaux R., M. Paskin and S. Madden: *Distributed Regression: an Efficient Framework for Modeling Sensor Network Data.* In Proceedings of the 3rd International Symposium on Information processing in Sensor Networks (IPSN '04), Berkeley, USA, April 2004.

[7] A. Jain and E.Y. Chang: *Adaptive Sampling for Sensor Networks.* In Proceeedings of the 1st International Workshop on Data Management for Sensor Net works (DMSN '04). Toronto, Canada, June 2004.

[8] A. Jain, E.Y. Chang and Y.-F. Wang: *Adaptive stream resource management using Kalman Filters.* In Proceedings of the ACM SIGMOD/PODS Conference (SIGMOD '04). Paris, France, June 2004.

[9] Y. Kotidis: *Snapshot Queries: Towards Data-Centric Sensor Networks.* In Proceedings of the 21st International Conference on Data Engineering (ICDE '05), Tokyo, Japan, April 2-5, 2005.

[10] S. Haykin: *Adaptive Filter Theory.* 4th ed., Prentice Hall, NJ, 2004.

[11] S. Haykin: *Least-Mean-Square Adaptive Filters.* Edited by S. Haykin, New York Wiley-Interscience, 2003.

[12] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan: *Energy-Efficient Communication Protocol for Wireless Microsensor Networks.* In Proceedings on the 33rd Hawaii International Conference on Computer Sciences (HICSS '00), Island of Maui, USA, Januray 2000.

[13] C. Intanagonwiwat, R. Govindan and D. Estrin: *Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks.* In Preoceedings of the 6th Intl. Conference on Mobile Computing and Networking (MobiCom '00), Boston, USA, August 2000.

[14] I. Lazaridis and S.Mehrotra: *Capturing Sensor-Generated Time Series with Quality Guarantee.* In Proceedings of the International Conference on Data Engineering (ICDE '03), Bangalore, India, March 2003.

[15] Y. Le Borgne and G. Bontempi: *Round Robin Cycle for Predictions in Wireless SensorNetworks.* In Proceedings of the 2nd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '05), Melbourne, Australia, December 2005.

[16] S.R. Madden, M.J. Franklin, J.M. Hellerstein and W. Hong: *TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks.* In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI '02), Boston, USA, December 2002.

[17] G. Moschytz and M. Hofbauer: *Adaptive Filter.* Springer Verlag, Berlin, 2000, ISBN 3-540-67651-1.

[18] C. Olston, B.T. Loo and J.Widom: *Adaptive precision setting for cached approximate values.* ACM SIGMOD, 2001.

[19] C. Olston and J.Widom: *Best effort cache synchronization with source cooperation.* In Proceedings of the ACM SIGMOD/PODS Conference (SIGMOD '02), Madison, USA, June 2002.

[20] A.V. Oppenheim and R.W. Schafer: *Discrete-Time Signal Processing.* Prentice Hall, 2nd edition, 1999.

[21] J. Polastre, R. Szewczyk and D. Culler: *Telos: Enabling Ultra-Low Power Research.* In Proceedings of the 4th International Symposium on Information processing in Sensor Networks (IPSN/SPOTS '05), April 2005.

[22] *http://db.lcs.mit.edu/labdata/labdata.html.*

[23] *http://www.intel-research.net/berkeley/index.asp.*

[24] *http://www.xbow.com/.*