Data Prediction in WSN using Variable Step Size LMS Algorithm

Biljana Stojkoska, Dimitar Solev, Danco Davcev
Faculty of Electrical Engineering and Information Technologies, University "Ss. Cyril and Methodius",
Skopje, Macedonia
biles@feit.ukim.edu.mk, dimitar.solev@gmail.com, etfdav@feit.ukim.edu.mk

***Abstract*— Wireless communication itself consumes the most amount of energy in a given WSN, so the most logical way to reduce the energy consumption is to reduce the number of radio transmissions. To address this issue, there have been developed *data reduction strategies* which reduce the amount of sent data by predicting the measured values both at the source and the sink, requiring transmission only if a certain reading differ by a given margin from the predicted values. While these strategies often provide great reduction in power consumption, they need a-priori knowledge of the explored domain in order to correctly model the expected values. Using a widely known mathematical apparatus called the Least Mean Square Algorithm (LMS), it is possible to get great energy savings while eliminating the need of former knowledge or any kind of modeling. In this paper with we use the Least Mean Square Algorithm with variable step size (LMS-VSS) parameter. By applying this algorithm on real-world data set with different WSN topologies, we achieved maximum data reduction of over 95%, while retaining a reasonably high precision.**

Keywords-Wireless Sensor Network; Data Prediction; Least Mean Square Algorithm; Time Series Forecasting.

I. INTRODUCTION

By being inherently distributed systems, WSN allow not only measuring the temporal progression of the ascertained quantity but also provide the ability to take the spatial progression of this quantity as well. By reporting data measurement at each interval, the node itself consumes a great deal of energy, thus, it vastly reduces its lifetime and creates sufficient communication overhead.

There are several techniques that have been developed to overcome these problems i.e., to lower the communication overhead, to increase energy saving and maximize CPU utilization.

Since wireless communication itself consumes the most amount of energy in a given WSN, the most logical way to reduce the energy consumption is to reduce the number of radio transmissions.

Data-reduction techniques aim to reduce the data to be delivered to the sink. These techniques can be divided into three main groups (Fig. 1): data compression, data prediction and in-network processing [1].

Data compression is applied to reduce the amount of information sent by source nodes. This scheme involves coding strategy used to represent data regardless of their semantics.

In-network processing performs data aggregation while data is routed towards the sink node. Data aggregation aims to transform the raw data into less voluminous refined data. It can be achieved with summarization functions (*minimum*, *maximum* and *average*). For applications that require original and accurate measurements, such a summarization may be inappropriate since it represents an accuracy loss [2].

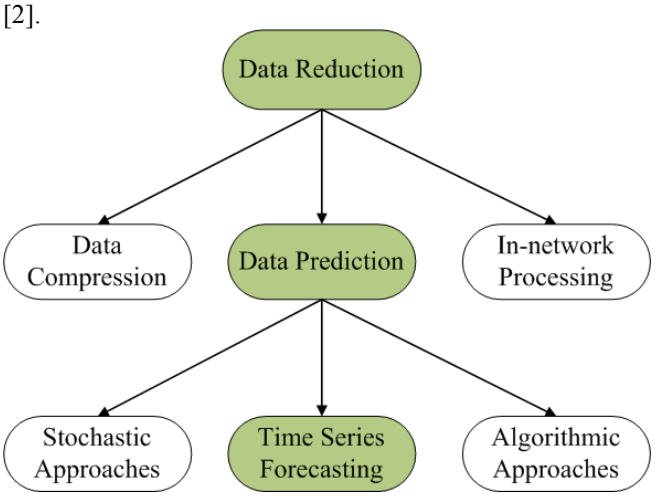

Fig. 1. Data-driven approach for energy saving in WSN [1].

Data prediction techniques usually maintain two instances of a model in the network, one residing at the sink and the other at the sensor. The model at the sink can be used to answer queries without requiring any communication, so the original data can be easily reconstructed within a certain degree of precision. 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.

Data prediction techniques can be divided into three subclasses: stochastic approaches, time series forecasting and algorithmic approaches, which are application specific (Fig. 1).

Stochastic approach is used when sensed phenomena can be considered as a random process by means of probability density function. Although this approach is general, its

 ISBN: 978-1-61208-144-1

computational overhead makes it inappropriate for tiny sensors with limited computational capacities.

The most appropriate models for data prediction in WSN are those based on time series forecasting. Moving Average (MA), Auto-Regressive (AR) or Auto-Regressive Moving Average (ARMA) models are simple, easy for implementation and provide acceptable accuracy [3][4]

In this paper, we investigated time-series forecasting technique for WSN based on LMS algorithm with variable step size (LMS-VSS). Least-Mean-Square (LMS) adaptive algorithm as a data-reduction strategy in WSN has been firstly proposed by Santini and Römer [5]. The advantages of LMS algorithm is that it does not require any a priori knowledge or modelling of the statistical properties of the observed signals, thus providing great flexibility and domain independence. Santini and Römer in [5] reported maximum data reduction of 92% for the temperature measurements (on Intel Berkeley Lab dataset [6]) while retaining an accuracy of 0.5°C.

The fact that a great percentage of the real-world wireless sensor networks are multi hop and hierarchy based, motivated us to exploit LMS-VSS on two different network topologies: star topology and cluster-based topology. The proposed algorithm is tested on real data obtained from the Intel Berkeley Research Laboratory sensor deployment [6]. LMS-VSS produce maximum data reduction of 95% for error margin of 0.5°C (for star topology) and around 97% when data aggregation was taken into account (cluster-based topology).

The rest of the paper is organized as follows: the next section explains least mean square algorithm. Section three of this paper describes the LMS-VSS. The fourth and the fifth section explore LMS-VSS on star network and cluster-based topologies respectively. Finally, we conclude this paper in section six.

II. LEAST MEAN SQUARE ALGORITHM

In this section, we present a brief explanation of the least mean square algorithm. A thorough explanation can be found in [6].

A linear adaptive filter samples a data stream/input signal at an instant n, which we will denote as $u[n]$ and calculates a prediction i.e., the output of the filter as $y[n] = \underline{w}^{T}[n]\cdot\underline{u}[n]$, which effectively is a linear combination of the previous N samples of the data stream (denoted as the vector $\underline{u}$ which is of length M), weighed by the corresponding weight vector $\underline{w}[n]$ (also of length M). M is an integer parameter that the filter uses and it determines the "memory" of the filter i.e., how many previous input sample it will use.

The output y[n] is then compared to the input signal or the sample of the data stream the filter tries to adapt to, denoted as d[n]. The prediction error e[n] is then computed as: e[n] = y[n]−d[n] and fed into the adaptation algorithm, so the filter weights can be updated. The vector w[n] i.e., the weights are modified at each time step n in order to minimize the mean square error.

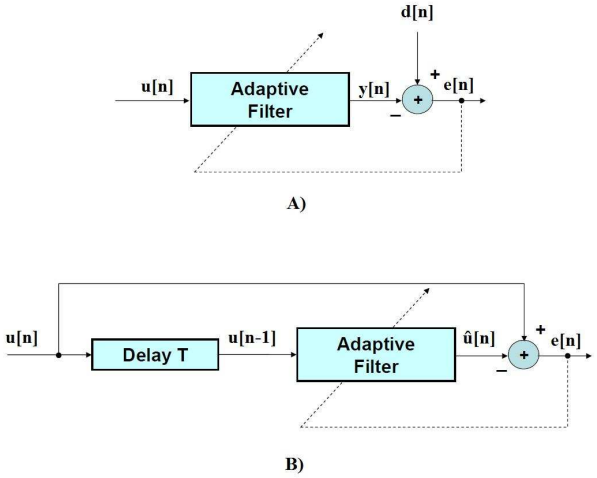

Fig. 2. A) Basic structure of an adaptive filter; B) Adaptive filter used for prediction.

One of the most extensively used adaptive algorithms is the Least-Mean-Square algorithm (LMS). Although it is extremely simple, it has quite good performances and has found implementation in a variety of applications. [7][8]. The LMS algorithm is defined through the three equations:

1. The filter output:
   $y[n] = \underline{w}^{T}[n]\cdot\underline{u}[n]$,
2. The estimation error:
   $e[n] = d[n] - y[n]$
3. The weight adaptation:
   $\underline{w}[n+1] = \underline{w}[n] + \mu\cdot\underline{u}[n]\cdot e[n]$

where $w[k]$ and $x[k]$ denote the M × 1 column vectors:

$$\underline{w}[k] = [w_1[k], w_2[k], \ldots, w_N[k]]^{T};$$
$$\underline{u}[k] = [u[k-1], u[k-2], \ldots, u[k-M]]^{T}.$$

With simple modification the filter structure from Fig. 2(A) to the so-called predictive structure of Fig. 2(B), the LMS algorithm can be used for prediction. Central to the successful prediction is delaying the current input value $u[n]$ by one time instance and use it as the reference signal $d[n]$. The filter then computes an estimation $\hat{u}[n]$ of the input signal at time instance n, as a linear combination of the N previous readings. Subtracting the prediction signal from the desired signal gives the value of the error which is fed back to adapt the filter weights.

Two parameters need to be defined for the adaptation process: the filter length M and the step-size parameter μ, which is important for updating the filter weights.

Given that $\underline{w}$ and $\underline{u}$ are M × 1 vectors, it can easily be concluded that the LMS algorithm requires $2M + 1$ multiplications and $2M$ additions per iteration. [7].

The practical implementation of the LMS algorithm in the prediction scheme in WSN was introduced in [5] where identical predictive filters were introduced both at the source and at the sink. LMS dual prediction scheme

(henceforth referred to as LMS-DPS) consists of simultaneously running an instance of the filter on both the node and the sink. There are three distinct modes of operation: initialization, normal and stand-alone. A node goes through the first only at the beginning and then switches between the normal and stand-alone modes. When a node is in the stand-alone mode it does not report its readings to the sink and this is where the energy savings are made.

The detailed description of each of the modes is as follows:

A. Initialization mode:

First, the step-size parameter μ must be determined. A certain amount of data must be collected in the beginning, so a proper estimation of the step-size can be made. At this time, the node keeps sending the data to the sink without making predictions. Both the node and the sink compute the value of μ. To ensure convergence [7][8] it must satisfy:

$$0 \le \mu \le \frac{2}{E_x} \tag{2}$$

where:

$$E_x = \frac{1}{N}\sum_{n=1}^{N}\left|x[n]\right|^2 \tag{3}$$

E_x is the mean power input and N is the number of iterations used to train the filter. Since the input mean power E_x is continually dependent on time, we can approximate $\hat{E}_x$ by computing over the first M samples i.e., N data readings and easily obtain the upper bound of (2).

After the initialization phase, both the node and the sink will continue to execute the predictive algorithm and node will be switching between the following two modes of operation.

B. Normal mode:

Both the node and the sink simultaneously execute the predictive algorithm and make a prediction for the following reading by using the last M readings and accordingly update the filter weights on the prediction error. As stated in [7], if no a priori knowledge is present, the initial weights should be zero. This is rather important, because:

a) we do not possess any a priori knowledge of the data;

b) setting the initial weights to zero (and using the same values for the step-size parameter and for the filter length), ensures that any instance of the LMS will behave exactly the same at any arbitrary point of time t_n.

The node will stay in normal mode (collecting data and reporting it to the sink) as long as the prediction error is greater than the maximum error budget e_{max}. When the error drops below e_{max} for M consecutive iterations, then the node switches to stand-alone mode i.e., stops reporting the readings and consequently stops updating the weights.

C. Stand-alone mode:

In this mode, the node still collects data and makes predictions, but if the prediction error is below e_{max}, instead of the reading $u[n]$, it feeds the filter with the prediction $y[n]$, discards the real reading $u[n]$ and does not send it to the sink. This enables both instances of the filter to be consistent and no update of the weights is needed (the error is zero) thus reducing the computational overhead.

If the prediction error exceeds e_{max} the node switches to normal mode and reports the reading. When the node is in this mode, the filter instance at the sink side uses only the predicted readings as an approximation of the real value.

III. LEAST MEAN SQUARE ALGORITHM WITH VARIABLE STEP SIZE

The step-size parameter μ is critical for the convergence of the algorithm i.e., it determines the convergence speed, so choosing the right value for μ is of critical importance [9]. As we explain later on, there are practical boundaries for the values that μ can receive and with a simple estimation technique, we can determine them. And as we show in this section, we introduce a specific improvement to the LMS-DPS algorithm regarding the step-size parameter μ.

Since LMS-DPS uses single value for the step-size μ, a further improvement can be made, with the introduction of a least mean square algorithm with variable step-size (which we will refer to as LMS-VSS henceforth), where the step-size parameter μ has two distinct values:

1. Until a certain number of good predictions are made, μ has the maximum value according to (3) and two orders of magnitude smaller to ensure robustness [8].
2. After μ has sufficiently learned what kind of data the filter receives, it switches to a stable value, that is:

$$\mu_{new} = \frac{\mu_{old}}{M}$$

where M is the filter length, and $\mu_{old} = 2 \cdot E_x^{-1} \cdot 10^{-2}$.

This improvement accelerates the initial adaptation to the data, so an additional 3-5% reduction can be gained where the maximum error is sufficiently small, as can be seen in Fig. 4.

A crucial aspect is when the switch should be made. The best value for the number of consecutive iterations in stand-alone mode can be found as:

$$M \le n \le M^2$$

where M is the filter length and n is the number of consecutive readings in stand-alone mode. The value $n=M^{3/2}$ yielded best results since it suited best both lower and higher filter lengths. For instance, if $M = 4$, using $n = M^2 = 16$ consecutive good predictions as a switch point is a good choice, but for $M = 10$, $n = M^2 = 100$ consecutive good predictions may never be reached, thus compromising performance. In this case, the choice $n = M = 10$ would be much more optimal. Also, note that the point of switching is determined such that both the node and sink can execute it

at the same time and thus retain consistency of both algorithm instances.

One of the advantages of having only two distinct values for the step-size parameter, one to accelerate the initial weight adaptation and another for fine-tuning the weights after the adaptation is that it contributes to the overall data reduction without creating additional computational overhead to the algorithm. Additionally, without using a specific heuristic for a particular type of data, it can be used in other schemes.

In order to compare LMS-VSS and LMS-DPS, both algorithms were implemented in MatLab. For algorithms evaluation, a set of experimental data from Intel Berkeley Research Lab network [6] was used. The 54 Mica2Dot sensors deployed in the laboratory were equipped with weather boards and measured humidity, temperature, light and voltage values once every 31 seconds. The measurements were collected between February 28th and April 5th, 2004. The dataset includes 2.3 million readings collected from these sensors. For our evaluation we use only temperature and humidity readings at different locations in a lab space. We run the simulations for 50 different error margins eMax (raging from 0.1°C to 5°C).

Metrics used for measuring algorithms performance varies from author to author. Some authors tend to reduce the number of transmissions, thus they count the number of sent massages from the sensor node to the sink node [5]. Here, the metric is the reduction of transmissions in percentage. Another way used for evaluating algorithm performance is by measuring the difference between the predicted and the true value, i.e., mean square error (MSE) or root mean square error (RMSE)[4]. In our case, the first metric is used, assuming that every transmission requires an equal amount of energy.

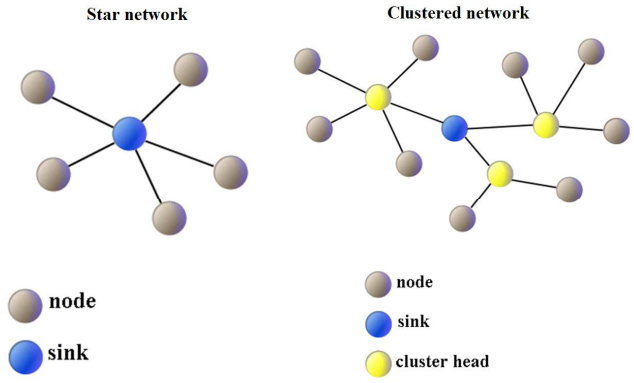

Fig. 3. Star network topology (left), Clustered-based network topology (right).

We consider two basic network topologies on which we evaluated the LMS-VSS algorithm (Fig. 3). The first is the star topology, where every node sends its readings directly to the sink. The advantage of this topology is its simplicity and low latency communications between the sensors and the sink, but the sink must be within radio transmission range of all the individual sensors. If this is not possible, cluster-based network topology is a suitable alternative.

IV. THE LMS-VSS ALGORITHM IN STAR TOPOLOGY

If a star network topology is considered, LMS-VSS algorithm achieves considerable reduction in number of transmissions. We use data readings from [6] and assume that each sensor sends its reading to the sink node. Fig. 4 shows the reduction gain when using LMS-DPS and LMS-VSS algorithms. The results are average for all 54 nodes from the Intel Berkeley Research Lab network [6]. LMS-DPS uses only one fixed step size μ=1.2·10^{-5} and filter length M=4 and obtain the average savings in the entire network of around 88% for error margin of 0.5°C. LMS-VSS uses the same filter length, but uses the first M=4 data readings to calculate the initial value of μ and then after $M^{3/2}$ readings switches to $\mu_{new} = \mu_{old} \cdot M^{-1}$. LMS-VSS obtain average savings in the entire network of around 92%.

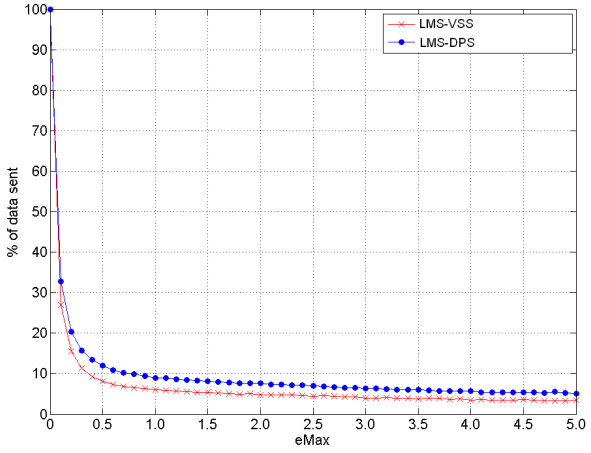

Fig. 4. The improvement of LMS-VSS over LMS-DPS algorithm for the entire Intel network (temperature).

In Fig. 5, we can see the improvement of LMS-VSS over the LMS-DPS algorithm for particular nodes. We chose these nodes because node 11 performs the best results (95% reduction for 0.5°C) and node 49 the worst (90% reduction for 0.5°C).

In addition, we investigated humidity readings from the same Intel network [6], where humidity is temperature corrected relative humidity, ranging from 0-100%. The results for node 20 and node 10 are shown on Fig. 6 and Fig. 7 respectively. As it can be seen from the figures, LMS-VSS performs more than 5% better results compared with LMS-DPS for error margin of 0.5°C, and more than 10% better results for error margin of 1°C and greater. For node 10, root mean square error is given for both algorithms (Fig.7). As can be seen from the figure, our algorithm gives smaller RMSE.

 ISBN: 978-1-61208-144-1

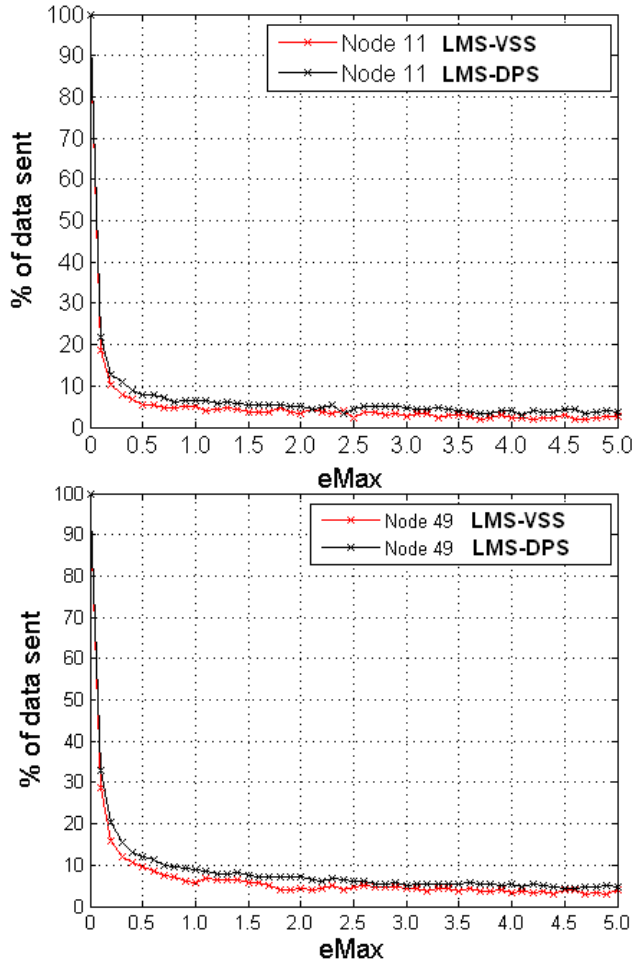

Fig. 5. Improvement of LMS-VSS over the LMS-DPS algorithm, filter length $M = 4$ (temperature).

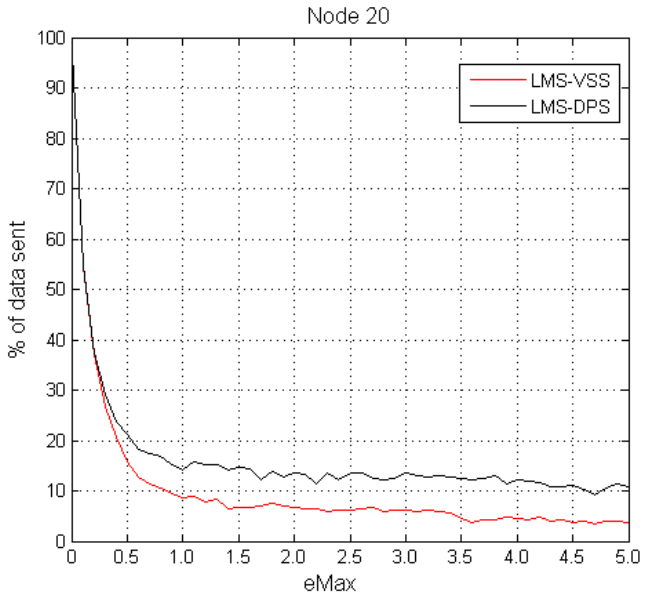

Fig. 6. Improvement of LMS-VSS over the LMS-DPS algorithm for humidity (node 20), filter length $M = 4$.

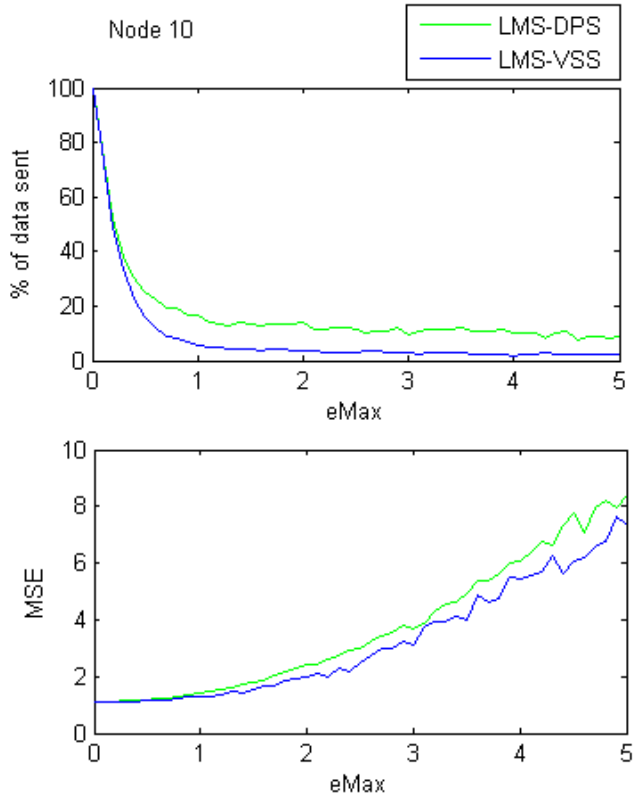

Fig. 7. Improvement of LMS-VSS over the LMS-DPS algorithm for humidity (node 10), filter length $M = 4$.

V. THE LMS-VSS ALGORITHM IN CLUSTER-BASED TOPOLOGY

Nodes clustering represents another way of prolonging WSN lifetime. Here, the sensor nodes are geographically grouped into clusters. In each cluster one representative node is chosen to be a cluster-head. Other nodes in the cluster are called members of the cluster and they report their readings to their cluster head, and cluster head forwards the messages directly to the sink. Note that except for the sink, we consider that all the nodes in the network have exactly the same characteristics, so even when a node is a cluster head, it still acts as a sensing node (and its readings are included in the computations for its respective cluster), with the additional duty to forward readings from cluster members. An example of such clustered-based network is given in [6], as presented on Fig. 8.

Our LMS-VSS algorithm was evaluated in this case by using MatLab simulation tool on the same Intel network [6]. We assume that each sensor sends its reading to the cluster head, and then cluster head resends the reading to the sink. As a result, each reading is sent twice, except the readings taken at the cluster heads. The clustering method we used is the simple k-means clustering, with k = 10. The value k = 10 was selected on a purely intuitive basis and it produced satisfactory results. Other clustering methods may be used as well. As it can be seen from Fig. 8, the clustering parameter was geographic position, i.e., Euclidian distance.

 ISBN: 978-1-61208-144-1

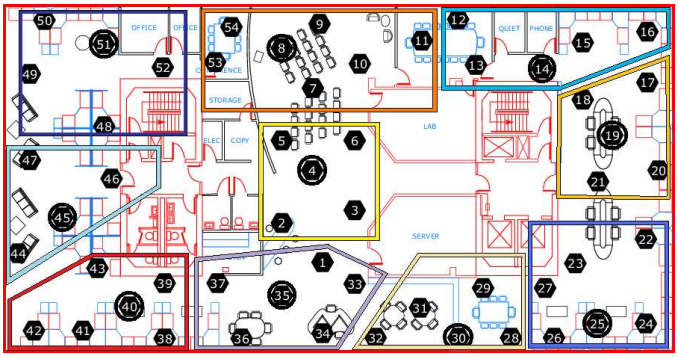

Fig. 8. A clustered view of the Intel Berkeley Research Lab wireless sensor network [6] (cluster heads are circled).

Additionally, we simulated data aggregation technique at cluster heads. In this scenario, every message carries exactly the same amount of data, since the cluster head aggregates the messages from its cluster members using a certain aggregation function (Average, Minimum, Maximum, etc) and forwards the aggregate to the sink.

The step size was $\mu = 1.2 \cdot 10^{-5}$ and the filter length was $M = 4$ for all the nodes, although changing the step-size or the length of the filter for a specific cluster can further improve the performance and the results.

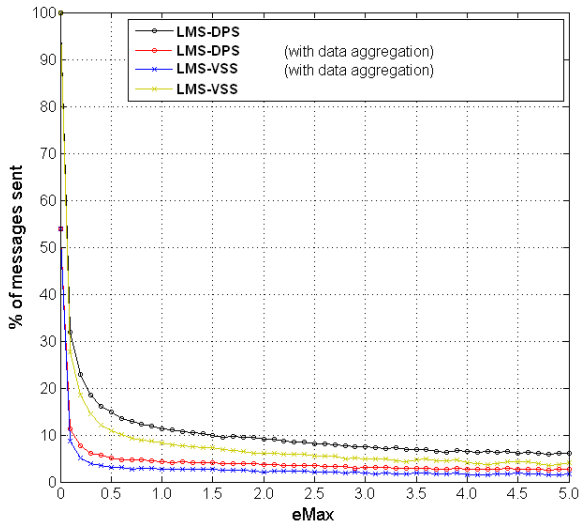

Fig. 9. Various savings in the cluster containing the nodes: 7, 8, 9, 10, 11, 53 and 54 (temperature).

We used the cluster containing nodes: 7, 8, 9, 10, 11, 53 and 54. As displayed in Fig. 9, the averages for this cluster show that there is a substantial gain of 5% when using the LMS-VSS algorithm over the LMS-DPM algorithm. When data aggregation is assumed to be implemented in the cluster, the reduction is far greater and when it is used alongside the LMS-VSS algorithm it results in 97% reduction of the total messages sent for the given error margin of 0.5C.

VI. CONCLUSION

In this paper, we investigated time-series forecasting technique for WSN based on LMS algorithm with variable step size (LMS-VSS). Santini and Römer in [5] reported maximum data reduction of 92% for the temperature measurements (on Intel dataset [6]) while retaining an accuracy of 0.5°C.

We exploited our LMS-VSS on two different network topologies: star topology and cluster-based topology. We evaluated the LMS-VSS on Intel dataset [6]. Our algorithm outperforms LMS-DPS in terms of data reduction and root mean square error for all evaluated nodes. LMS-VSS performs data reduction of around 92% for error margin of 0.5°C (for the entire Intel network using star topology), and maximum data reduction of around 95% for particular nodes. In cluster-based topology, when data aggregation was taken into account LMS-VSS achieves data reduction of around 97%.

From the simulated results, we can conclude that star network is the most suitable network topology by means of energy saving, since each reading is sent only once. If sensors are not within each other radio range, cluster-based topology could be used, where each reading is being resend by the cluster head. By applying data aggregation at cluster head, this topology can achieve even greater data reduction in scenarios where loosing data precision is affordable. It is obvious that the trade-off is application specific.

For future work, we intend to investigate our LMS-VSS on tree-based network topology, and to evaluate the results using different datasets (from Intel and other networks). We also plan to explore machine learning techniques for choosing the best values for the parameter μ or implement a scheme that will dynamically readjust the filter length.

REFERENCES

[1] G. Anastasi , M. Conti , M. Di Francesco , and A. Passarella, Energy conservation in wireless sensor networks: A survey, Ad Hoc Networks, vol.7 n.3, pp. 537-568, May, 2009.

[2] E. F. Nakamura , A. A. F. Loureiro , and A. C. Frery, Information fusion for wireless sensor networks: Methods, models, and classifications, ACM Computing Surveys (CSUR), vol.39 n.3, pp. 9-55, 2007.

[3]Y. L. Borgne , S. Santini , and G. Bontempi, Adaptive model selection for time series prediction in wireless sensor networks, Signal Processing, vol.87 n.12, pp. 3010-3020, December, 2007.

[4] C. Liu, K. Wu, and M. Tsao, "Energy Efficient Information Collection with the ARIMA Model in Wireless Sensor Networks", In Proceedings from IEEE Globecom, vol. 5, pp. 2470–2474, 2005.

[5] S. Santini and K. Römer: An Adaptive Strategy for Quality-Based Data Reduction in Wireless Sensor Networks, Proceedings of the 3rd International Conference on Networked Sensing Systems (INSS 2006), pp. 29-36, Chicago, IL, USA. June 2006.

[6] Intel lab data. Web Page, (Accessed on 06/06/2011) http://db.lcs.mit.edu/labdata/labdata.html

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

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

[9] R. Kwong and E.W. Johnston, A Variable Step Size LMS Algorithm, IEEE Trans. On Signal Process., vol. 40, pp. 1633 - 1642, 1992.