

An Improved Adaptive Signal Processing Approach To Reduce Energy Consumption in Sensor Networks

Madhavi L. Chebolu, Vijay K. Veeramachaneni, Sudharman K. Jayaweera and Kameswara R. Namuduri
Department of Electrical and Computer Engineering
Wichita State University
Wichita, KS, 67226, USA.
Email: mlchebolu, vkveeramachaneni@wichita.edu

Abstract—An improved distributed coding and signal processing approach is proposed to reduce energy consumption in wireless sensor networks. The proposed scheme exploits the inherent correlations among sensor data in order to reduce the transmission requirements. The energy consumption at data collection sensor nodes is reduced by implementing the correlation tracking algorithm at a data gathering node. The proposed algorithm extends the scheme proposed in [1] by improving the adaptive correlation tracking algorithm. Our numerical results based on sample real sensor data show that the proposed scheme can provide significant energy savings.

I. INTRODUCTION

Large-scale wireless sensor networks can be used in many applications such as, in monitoring audio, video, temperature, or pressure. These sensor networks consist of sensing nodes which are battery powered. The energy aware sensor processing assumes an important role in such sensor networks in increasing the network life-time and preventing the loss of critical information. The major causes of energy expenditure by sensors in wireless networks are transmission and reception of data. Thus, energy efficient techniques that reduce the number of sensor transmissions are an important area of research in wireless sensor network design.

There are various approaches that can be of help in reducing energy consumption in sensor networks. For example, one of the approach is to switch off the nodes that are not transmitting or receiving the data conserving the battery power at nodes [2]. Another example is to use EC-MAC (Energy Conserving - Medium Access Control) protocol which utilizes a centralized scheduler avoiding the collisions over wireless channel and reducing the number of retransmissions [3]. Other methods of efficient information processing to reduce energy consumption can be found in [4] and [5].

Usually the data from different sensors exhibit spatio-temporal correlations. A good example is the sensor data involving audio field sensors, as they pick up attenuated and delayed versions of a signal from a common audio source. In [1], a scheme that exploits the redundancy caused by these inherent correlations in order to reduce energy consumption was proposed. This scheme is useful for network models consisting of two types of nodes: sensing nodes and a data gathering node. The sensing nodes are assumed to have limited

battery power while the data gathering node is assumed to have unlimited energy. The control is assumed to be centralized at the data gathering node. The sensors compress their data with respect to the other sensor data but without knowing the existing correlation structure, before sending it to the data gathering node. In this approach energy efficiency is achieved by reducing the inter-sensor communications.

The approach proposed in [1] consisted of a computationally inexpensive encoder algorithm supporting multiple compression rates and an adaptive correlation-tracking algorithm. The distributed compression technique used for encoding compresses the sensed data based on the information about the correlation structure among sensor data provided by the data gathering node. The correlation tracking algorithm at the data gathering node employs a linear predictive model which linearly combines the values available at the data gathering node in order to predict the current observation of a particular sensor. Those values include the past readings of the particular sensor for which the present value is being predicted and the present values of the sensor readings available or already decoded at the data gathering node. This linear predictive model adaptively determines the correlation structure of the data being sensed by all the data collection sensors. This correlation information is provided to the sensors so as to help them in compressing data.

In this paper we propose an extension to the basic scheme developed in [1] by improving the performance of the above linear predictive model. We achieve this by modifying the linear predictive model to include the history of the other sensors along with their current readings. Our simulation results show that by including the past readings in the prediction model it is possible to obtain an improvement of 5%-10% in energy savings.

In section II details of the algorithms employed at both sensors and the data gathering node are given. The details of the distributed compression technique at sensors are explained followed by the details of correlation tracking algorithm. Simulation results are given in Section III followed by concluding remarks in Section IV.

II. SYSTEM MODEL AND DESCRIPTION

In a sensor network the sensors need to transmit, receive and process data. In an energy-constrained sensor network, one way to reduce the energy consumption of the sensors is to reduce the energy spent for transmitting and receiving data. This can be achieved by encoding or compressing the data by exploiting the redundancy due to inherent spatio-temporal correlations. But in order to achieve this type of compression all sensors need to keep track of the existing sensor data correlations requiring inter-sensor communication which increases the energy consumption.

In order to avoid inter-sensor communication, in [1] the control was centralized at a data-gathering node that keeps track of the correlation structure among different sensor data. Sensors encode their readings based on the information provided by the data-gathering node avoiding the need for inter-sensor communication. Thus, the complexity of the system is transferred to the data-gathering node. This allows the use of lightweight encoders that uses a distributed compression algorithm at the sensor nodes resulting in significant energy savings at the sensors. An adaptive prediction algorithm whose function is to keep track of the existing correlation structure is employed at the data-gathering node. In the following we describe each of these in detail.

A. Distributed Compression / Encoding Algorithm at Sensor nodes

The encoding algorithm based on the Slepian-Wolf coding theorem [6] allows each sensor to compress its data without knowing other sensor measurements or its data correlation with other sensors' data.

In the proposed architecture, one sensor called the reference sensor sends its data either uncoded, or compressed with respect to its own past, to the data gathering node. Based on the information provided by the data gathering node about the reference sensor's data, the second sensor compresses its data with respect to its own past and the first sensor's data. The compression rate depends on the amount of correlation in the data. This architecture can also be generalized to networks with multiple data gathering nodes.

The code construction for distributed algorithm should support multiple compression rates and it should be computationally inexpensive. Based on these requirements a tree-based distributed compression algorithm is devised which uses an underlying codebook common to all the sensors as well as the data gathering node.

The codebook construction starts with a root codebook of 2^n representative values of an n -bit A/D converter and partitioned into two subsets of even-indexed and odd-indexed in a chained fashion which results in an n -level tree structure. The data gathering node requests data from a sensor and it provides information about correlation structure among the sensors indirectly, i.e., it informs the sensor the number of bits i the sensor needs to use to represent its data. This i represents the level of the codebook tree to which the subcodebook containing actual data belongs to. In the i -th

level, the subcodebook has values spaced $2^i \Delta$ apart where Δ is the precision of the n -bit A/D converter. So a sensor needs to use only i bits instead of the n bits, where $i < n$, in order to represent the output of the A/D converter, if the sensor reading and the side information at the data-gathering node are no further apart than $2^{i-1} \Delta$. The side information is obtained by exploiting the correlation structure among sensor data that is known to the data-gathering node. The data-gathering node computes the required value of i using the correlation structure of the sensor readings. For more details of the encoding algorithm we refer the reader to [1].

B. Adaptive Correlation-Tracking and Decoding Algorithms at the Data-gathering node

First the data gathering node makes queries for direct readings (uncoded) from all the sensors for a training duration of T rounds. This training is required so as to find the correlation structure that exists among the sensor data before starting the compression of real time data. After the training period, the data gathering node starts adaptive correlation tracking and the sensors start to send compressed readings. To reduce the reception energy of sensors i , or the number of bits required to compress the sensor readings, are only sent periodically once in every K samples. The value of K must be chosen carefully as it effects the decoding errors. In our simulations we have set $K = 100$.

1) *Correlation Tracking Algorithm:* The data-gathering node computes the number of bits i used by each sensor to compress its observation. The data gathering node computes this value of i based on the tracked correlation structure among the sensor data. To decode the compressed reading from the j -th sensor, it needs to compute side-information for the j -th sensor reading at k -th time instant.

The linear predictive model is based on a linear combination of the past values of the current sensor for which we are finding the side-information and current and past readings of all other sensors which are available at the data gathering node:

$$Y_k^{(j)} = \sum_{l=1}^{M_1} \alpha_l X_{k-l}^j + \sum_{i=1}^{j-1} \beta_i X_k^i + \sum_{i=1}^{j-1} \sum_{l=1}^{M_2} \theta_l^i X_{k-l}^i, \quad (1)$$

where X_{k-l}^j represents the past readings of sensor j , X_k^i represents the present reading of sensor i and $Y_k^{(j)}$ is the side information for the j -th sensor data computed at the decoder at time instant k . The prediction $Y_k^{(j)}$ determines the number of bits needed to represent X_k^j . Note that, M_1 and M_2 are the number of past values of the sensor j and of already decoded sensors used in the prediction model, respectively.

The linear predictive model in [1] uses the past values of the sensor for which the side information is being calculated and the current readings of different sensors which are available at the data gathering node. In our proposed model (1) we consider the effect of including the past values of the reference sensor (along with its present) and all other sensors current readings along with their past readings (either decoded or direct readings) available at the data gathering node at

that instant. Depending on the length of the past readings included, the information about the amount of correlation among the sensor data varies which effects the compression rate. Generally, better compression rate with less decoding errors are achieved by accurate correlation tracking provided by increasing the length of the past readings in the model (1). This effects the number of bits required for compression by the sensors. The lesser the number of bits used for data representation the more the energy savings achieved by the sensors as the energy used in transmitting bits to the data gathering node reduces.

We choose the coefficients α_l , β_i and θ_l^i in (1) to minimize the mean squared prediction error given by:

$$E[N_j^2] = r_{x^j x^j}(0) - 2\vec{P}_j^T \vec{\Gamma}_j + \vec{\Gamma}_j^T R_{ZZ}^j \vec{\Gamma}_j, \quad (2)$$

where $N_j = Y_k^{(j)} - X_k^{(j)}$.

In (2) $\vec{\Gamma}_j$ is a column vector of α 's, β 's and θ 's and we have we assumed that X_k^j and X_k^i are pairwise jointly wide-sense stationary. Note that R_{ZZ}^j can be written as:

$$R_{ZZ}^j = \begin{bmatrix} R_{x^j x^j} & R_{x^j x^i} & R'_{x^j x^i} \\ R_{x^j x^i}^T & R_{x^i x^i} & R'_{x^i x^i} \\ R'_{x^j x^i} & R'_{x^i x^i} & R'_{x^i x^i} \end{bmatrix},$$

$$R_{x^j x^j} = \begin{bmatrix} r_{x^j x^j}(0) & r_{x^j x^j}(1) \dots r_{x^j x^j}(M_1 - 1) \\ r_{x^j x^j}(1) & r_{x^j x^j}(0) \dots r_{x^j x^j}(M_1 - 2) \\ \dots & \dots \\ r_{x^j x^j}(M_1 - 1) & r_{x^j x^j}(M_1 - 2) \dots r_{x^j x^j}(0) \end{bmatrix},$$

$$R_{x^j x^i} = \begin{bmatrix} r_{x^j x^1}(1) & r_{x^j x^2}(1) & \dots & r_{x^j x^{j-1}}(1) \\ r_{x^j x^1}(2) & r_{x^j x^2}(2) & \dots & r_{x^j x^{j-1}}(2) \\ \dots & \dots & \dots & \dots \\ r_{x^j x^1}(M_2) & r_{x^j x^2}(M_2) & \dots & r_{x^j x^{j-1}}(M_2) \end{bmatrix},$$

$$R_{x^i x^i} = \begin{bmatrix} r_{x^1 x^1}(0) & r_{x^1 x^2}(0) & \dots & r_{x^1 x^{j-1}}(0) \\ r_{x^2 x^1}(0) & r_{x^2 x^2}(0) & \dots & r_{x^2 x^{j-1}}(0) \\ \dots & \dots & \dots & \dots \\ r_{x^{j-1} x^1}(0) & r_{x^{j-1} x^2}(0) & \dots & r_{x^{j-1} x^{j-1}}(0) \end{bmatrix},$$

$$R'_{x^j x^i} = [R_{x^j x^1} \quad R_{x^j x^2} \quad R_{x^j x^3} \quad \dots \quad R_{x^j x^{j-1}}]^T,$$

$$R'_{x^i x^i} = [R_{x^1 x^i} \quad R_{x^2 x^i} \quad R_{x^3 x^i} \quad \dots \quad R_{x^{j-1} x^i}]^T,$$

$$R'_{x^i x^i} = \begin{bmatrix} R_{x^1 x^1} & R_{x^1 x^2} & \dots & R_{x^1 x^{j-1}} \\ R_{x^2 x^1} & R_{x^2 x^2} & \dots & R_{x^2 x^{j-1}} \\ \dots & \dots & \dots & \dots \\ R_{x^{j-1} x^1} & R_{x^{j-1} x^2} & \dots & R_{x^{j-1} x^{j-1}} \end{bmatrix},$$

$$\vec{\Gamma}_j = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_{M_1} \\ \beta_1 \\ \beta_2 \\ \dots \\ \beta_{j-1} \\ \theta_1^1 \\ \theta_2^1 \\ \dots \\ \theta_{M_2}^1 \\ \theta_1^2 \\ \theta_2^2 \\ \dots \\ \theta_{M_2}^2 \\ \dots \\ \theta_1^{j-1} \\ \theta_2^{j-1} \\ \dots \\ \theta_{M_2}^{j-1} \end{pmatrix}, \quad \vec{P}_j = \begin{pmatrix} r_{x^j x^j}(1) \\ r_{x^j x^j}(2) \\ \dots \\ r_{x^j x^j}(M_1) \\ r_{x^1 x^j}(0) \\ r_{x^2 x^j}(0) \\ \dots \\ r_{x^{j-1} x^j}(0) \\ r_{x^1 x^j}(1) \\ r_{x^1 x^j}(2) \\ \dots \\ r_{x^1 x^j}(M_2) \\ r_{x^2 x^j}(1) \\ r_{x^2 x^j}(2) \\ \dots \\ r_{x^2 x^j}(M_2) \\ \dots \\ r_{x^{j-1} x^j}(1) \\ r_{x^{j-1} x^j}(2) \\ \dots \\ r_{x^{j-1} x^j}(M_2) \end{pmatrix},$$

and

$$r_{x^j x^i}(l) = E[X_k^j X_{k+l}^i].$$

In order to choose the set of coefficients that minimizes the mean squared error differentiate the mean squared error $E[N_j^2]$ with respect to $\vec{\Gamma}_j$. Then we obtain

$$\frac{\partial E[N_j^2]}{\partial \vec{\Gamma}_j} = -2\vec{P}_j + 2R_{ZZ}^j \vec{\Gamma}_j. \quad (3)$$

Thus, the MMSE optimal $\vec{\Gamma}_j$ is given by

$$\vec{\Gamma}_{j,opt} = R_{ZZ}^{-1,j} \vec{P}_j. \quad (4)$$

2) *Least-Mean-Squares (LMS) Algorithm:* During the first T rounds the data-gathering node requests for the direct readings from all the sensors in order to adaptively compute the above estimate of $\vec{\Gamma}_{j,opt}$. Least-mean-squares algorithm [7] is used for adaptively updating the parameter vector based on the prediction error which is the difference between the side-information and the decoded reading at the same instant of time. The coefficient vector is updated using the following LMS algorithm:

$$\vec{\Gamma}_j^{(k+1)} = \vec{\Gamma}_j^{(k)} + \mu \vec{Z}_{k,j} N_{k,j}, \quad (5)$$

where μ is the step size and

$$\vec{Z}_{k,j} = \begin{pmatrix} X_{k-1}^{(j)} \\ X_{k-2}^{(j)} \\ \dots \\ X_{k-M_1}^{(j)} \\ X_k^{(1)} \\ X_k^{(2)} \\ \dots \\ X_k^{(j-1)} \\ X_{k-1}^{(1)} \\ X_{k-2}^{(1)} \\ \dots \\ X_{k-M_2}^{(1)} \\ X_{k-1}^{(2)} \\ X_{k-2}^{(2)} \\ \dots \\ X_{k-M_2}^{(2)} \\ \dots \\ X_{k-1}^{(j-1)} \\ X_{k-2}^{(j-1)} \\ \dots \\ X_{k-M_2}^{(j-1)} \end{pmatrix}.$$

The main steps involved in the LMS algorithm are:

- $Y_k^{(j)} = \vec{\Gamma}_j^{(k)T} \vec{Z}_{k,j}$
- $N_{k,j} = X_k^{(j)} - Y_k^{(j)}$
- $\vec{\Gamma}_j^{(k+1)} = \vec{\Gamma}_j^{(k)} + \mu \vec{Z}_{k,j} N_{k,j}$

3) *Decoding*: The data gathering node decodes the compressed readings sent by the sensors. As mentioned earlier the underlying codebook is common to all the sensors and the data gathering node. So the compressed reading from sensor j which is coded with respect to its corresponding side information $Y_k^{(j)} = \vec{\Gamma}_j^{(k)T} \vec{Z}_{k,j}$ using i bits, is decoded to the closest codeword in the subcodebook \mathcal{S} at level i in the underlying codebook as

$$\hat{X}_k^{(j)} = \arg \min_{r_i \in \mathcal{S}} \| Y_k^{(j)} - r_i \|. \quad (6)$$

As mentioned previously, at the i -th level the subcodebook entries are spaced $2^i \Delta$ apart. Thus, if the difference between the actual reading and the prediction is less than $2^{i-1} \Delta$, then $\hat{X}_k^{(j)}$ will be decoded correctly. But if $N_{k,j} > 2^{i-1} \Delta$ then decoding errors will occur. This probability of decoding error can be bounded by using the Chebyshev's inequality [8] as

$$P[|N_{k,j}| > 2^{i-1} \Delta] \leq \frac{\sigma_{N_j}^2}{(2^{i-1} \Delta)^2}. \quad (7)$$

where the random variable $N_{k,j}$ is assumed to have a distribution with zero mean and variance $\sigma_{N_{k,j}}^2$. If we want the probability of error to be less than a given value P_e , then

$$P_e = \frac{\sigma_{N_j}^2}{(2^{i-1} \Delta)^2}. \quad (8)$$

The number of bits required for compression by a specific sensor is calculated from (8) as

$$i = \frac{1}{2} \log_2 \left(\frac{\sigma_{N_j}^2}{\Delta^2 P_e} \right) + 1. \quad (9)$$

In order to calculate i as in (9), the data gathering node needs to estimate the variance of the prediction error. At the end of T rounds of training the data gathering node initializes the estimate of error variance $\sigma_{N_j}^2$ as,

$$\sigma_{N_j}^2 = \frac{1}{T-1} \sum_{k=1}^T N_{k,j}^2. \quad (10)$$

After the training period due to the time-varying nature of the sensor data the variance needs to be updated. This updating is performed using the following filtered estimate:

$$\sigma_{N_j, new}^2 = (1 - \gamma) \sigma_{N_j, old}^2 + \gamma N_{k,j}^2. \quad (11)$$

where $0 \leq \gamma \leq 1$ is a *forgetting factor* [7].

III. NUMERICAL RESULTS

Simulations are performed for the sensor data sets of humidity, temperature and light that were used in [1]. In our sensor network, we considered 5 data collection sensors and 1 data gathering node. We simulated our proposed algorithm over 5 sensors of humidity data-set with 17765 samples for each sensor, over 5 sensors of temperature data-set with 17764 samples for each sensor and over 4 sensors of light data-set with 17653 samples for each sensor. We assumed a 12-bit A/D converter with a dynamic range of $[-128, 128]$ for humidity and temperature data sets and a dynamic range of $[-256, 256]$ for light. The prediction of the reading for sensor j is derived as

$$Y_k^{(j)} = \sum_{l=1}^{M_1} \alpha_l X_{k-l}^j + \sum_{i=0}^{M_2} \theta_i X_{k-i}^{(reference.sensor)}. \quad (12)$$

That is the prediction for a particular sensor j is the linear combination of the M_1 past readings of sensor j and the current reading of the reference sensor along with its M_2 past readings ($X_{k-i}^{(reference.sensor)}$). α_l 's and θ_i 's are the weighting coefficients. The reference sensor can be one of the $j-1$ sensors that have been already decoded. (Note that, we assume the sensors are labelled according to the order they are decoded.)

Correlation tracking algorithm is tested as in [1] by calculating the tolerable noise at each time instant and comparing it with the actual prediction noise. Note that the amount of noise that can exist between the prediction of a sensor and the actual reading without including a decoding error is called the tolerable noise. Thus tolerable noise is given by $2^{i-1} \Delta$. Typical plots of tolerable noise vs. actual prediction noise are given in Fig.1. The top graph of each plot shows the tolerable noise and bottom graph represents the prediction noise. It can be observed from these plots that the tolerable noise is well above the actual noise. The maximum tolerable noise is 128 (not shown in the graph). The spikes in the tolerable noise

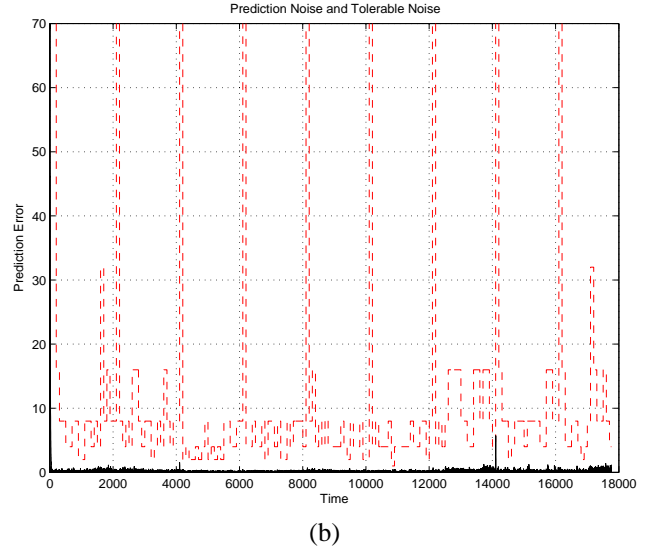
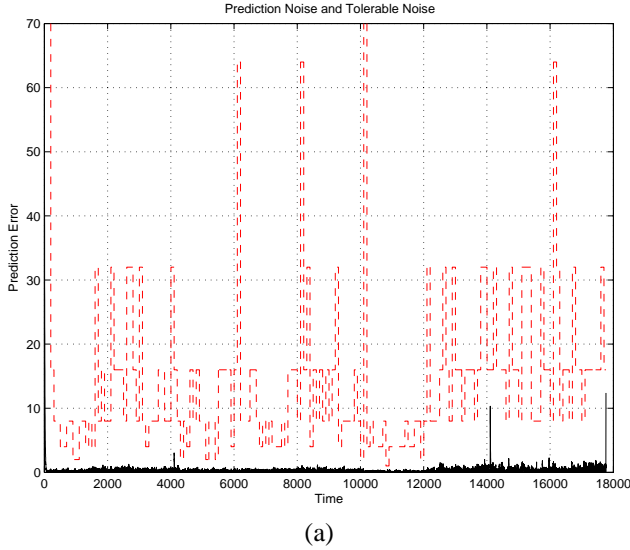


Fig. 1. Tolerable Noise Vs. Prediction Noise for Humidity Data ($P_e = 0.001, \gamma=0.5, \mu = 10^{-5}, T = 100$ and Zero Decoding Errors) (a) Algorithm of [1] (Energy Savings=30.37%). (b) Proposed Algorithm (Energy Savings=35.48%).

are due to the aggressive forgetting factor chosen in (11). By choosing a proper value for the forgetting factor we can reduce these spikes and obtain a tradeoff between decoding errors and energy savings. An extra $\log(n)$ bits are used by each sensor to receive information about the value of i from data gathering node. Note that, we have assumed that energy used by a sensor to transmit a bit is equivalent to the energy used to receive a bit.

We tried to optimize the parameter set used in LMS algorithm for the algorithm in [1]. The best results we were able to obtain show only about 31% of energy savings with zero decoding errors for humidity data set. We compared our results with respect to the results produced by the scheme in [1] for this optimized set of parameters. Tables of results, comparing the proposed algorithm and the algorithm of [1] in terms of percentage of energy savings, percentage of decoding errors, with different values for M_1 and M_2 in the linear predictive model for the humidity data set are given below (Note that, rows labelled as A and B correspond to results obtained with the algorithm of [1] and results obtained with the proposed algorithm, respectively):

TABLE I

COMPARISON OF ENERGY SAVINGS AND DECODING ERRORS FOR THE PROPOSED ALGORITHM AND THE ALGORITHM IN [1] WITH $P_e = 0.001, \gamma=0.5, \mu = 10^{-5}, T = 100$ (FOR ZERO DECODING ERRORS CASE)

	M_1	M_2	%of Energy savings	%of decoding errors
A	4	0	30.37	0
B	4	4	35.48	0
A	16	0	32.42	0
B	16	16	37.79	0

From Table I we observe that the proposed algorithm offers about 5% of energy savings compared to that of [1]. Also, we can observe from these results that increasing M_1 in general

TABLE II

COMPARISON OF ENERGY SAVINGS AND DECODING ERRORS FOR THE PROPOSED ALGORITHM AND THE ALGORITHM IN [1] WITH $P_e = 0.01, \gamma=0.4, \mu = 10^{-5}, T = 100$

	M_1	M_2	%of Energy savings	%of decoding errors
A	4	0	41.41	1.5
B	4	4	46.89	0.99
A	16	0	43.764	0.9
B	16	16	49.72	0.4

TABLE III

COMPARISON OF ENERGY SAVINGS AND DECODING ERRORS FOR THE PROPOSED ALGORITHM AND THE ALGORITHM IN [1] WITH $P_e = 0.02, \gamma=0.2, \mu = 10^{-5}, T = 100$

	M_1	M_2	%of Energy savings	%of decoding errors
A	4	0	43.81	4.3
B	4	4	51.39	0.91
A	16	0	46.43	4.2
B	16	16	54.81	0.3

increases the energy savings.

We can further increase the energy saving figures in Table-I if we are willing to tolerate some decoding errors. For example, in Table II we have shown energy savings for a particular parameter set that results in some decoding errors.

Comparing Table I and Table II, we can see that with $(M_1, M_2) = (4, 4)$ and $(M_1, M_2) = (16, 16)$ of B, 11% and 12% of extra energy savings is achieved respectively, by the proposed algorithm. By comparing Table I and Table III we can see that with $(M_1, M_2) = (4, 4)$ and $(M_1, M_2) = (16, 16)$ of B, 16% and 17% of extra energy savings is achieved, respectively.

From Table III we observe that for humidity data set there is an improvement of about 8% from A to B with $(M_1, M_2) = (4, 4)$ and $(M_1, M_2) = (16, 16)$ of B, if we were to use the

proposed algorithm compared to that of algorithm in [1]. These results show that energy savings increase by increasing M_2 in the linear predictive model.

In practice depending on the requirements we can establish a trade-off between decoding errors and energy savings and optimize the parameters accordingly. For example, in some specific cases the energy savings may be more important than minimizing decoding errors. In such cases we can improve the system life-time by choosing a relatively large value for M_2 . But in cases where the information provided by the sensors is critical error-free decoding of sensor readings may be preferred over the energy savings. In either case, the proposed algorithm can offer better performance compared to that of [1].

IV. CONCLUSIONS

In this paper we improved the method proposed in [1] for reducing the energy consumption in sensor networks using adaptive prediction technique by including the past readings of other sensors in the linear predictive model at the data gathering node. The proposed modification results in better compression at the sensor level. This efficiency combined with the robustness of the model provide significant energy savings and lower decoding error rates in practice as confirmed by our numerical results.

REFERENCES

- [1] J. Chou, D. Petrovic, and K. Ramchandran, "A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks," in *IEEE INFOCOM*, San Francisco, CA, Mar. 2003.
- [2] C. S. Raghavendra and S. Singh, "PAMAS - Power Aware Multi-Access Protocol with Signalling for Ad hoc Networks," in *Computer Communications Review*, July 1998.
- [3] C. E. Jones, K. M. Sivalingam, P. Agarwal, and J. C. Chen, "A survey of energy efficient network protocols for wireless networks," in *Wireless Networks*, ser. 4, vol. 7, July 2001, pp. 343 – 358.
- [4] G. Pottie and W. Kaiser, "Wireless sensor networks," in *Communications of the ACM*, 2000.
- [5] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," in *IEEE Journal of High Performance Computing Applications*, 2002.
- [6] D. Slepian and J. K. Wolf, "Noiseless encoding of correlated information sources," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 471–480, July 1973.
- [7] S. Haykin, *Adaptive Filter Theory*. Upper-saddle River, NJ, USA: Prentice Hall, 1996.
- [8] H. Stark and J. Woods, *Probability, Random Processes and Estimation Theory for Engineers*. Englewood Cliffs: Prentice Hall, 1994.