2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing

A Decentralized Quality Aware Adaptive Sampling Strategy in Wireless Sensor Networks

A. Masoum, N. Meratnia, P. J. M. Havinga
Pervasive Systems, Department of Computer Science University of Twente, The Netherlands
{a.masoum, n.meratnia, p.j.m.havinga} @utwente.nl

***Abstract*—Since WSNs suffer from sever resource constraints, in terms of energy, memory and processing, temporal, spatial and spatio-temporal correlation among sensor data can be exploited by adaptive sampling approaches to find out an optimal sampling strategy, which reduces the number of sampling nodes and/or sampling rates while maintaining high data quality. In this paper, a quality aware decentralized adaptive sampling strategy is proposed which benefit from the data correlation for predicting future samples. In this algorithm, sensor nodes adjust their sampling rates, based on environmental conditions and user defined data range. Simulation results show that proposed approach provides 90 percentage event detection accuracy level while consumes lesser energy rather than existing adaptive sampling approach.**

I. INTRODUCTION

WIRELESS sensor networks (WSNs) are new revolutionary monitoring platforms. Their high dynamicity and sever resource constrains necessitate optimal resource management policy. Such a policy dictates the quality and quantity of sensor data and tasks executed by sensor nodes. Resource management aims to answer the following question: Given an application structure, its quality of service (QoS) requirements, and system's current state, how to manage and allocate restricted resources, i.e., sensing, processing and communication resources, to achieve the best performance [1].

Some efforts in resource management have been directed towards finding optimal routing algorithms or data aggregation methods, which aim to achieve an efficient resource allocation by taking into account latency, throughput, data quality, and energy as QoS parameters. Dealing with sampling policies is another possible approach for resource management. Determining which sensor nodes and how often should collect data in such a way that application quality of service requirements are satisfied is the challenge faced by resource management solutions. To do so, exploring data correlation is a promising concept.

One of the most important QoS parameters in WSN applications is data quality, which implies representativeness of the data collected at the base station compared with the situation of monitored phenomena. It is somewhat clear that ensuring high data quality does not always go hand in hand with ensuring economical usage of resources. Usually, ensuring data quality comes at the expense of energy expenditure [2].

In order to capture dynamic changes of the monitored phenomena, on the one hand, and to save energy, on the other, having fixed sampling plans should be avoided. This requires having an adaptive sampling strategy in place, which can be configured and adjusted according to the dynamicity observed.

One of these adaptive sampling approaches is proposed by Chatterja and Havinga [7]. In this approach, each sensor node considers temporal correlation among readings to adjust its sampling rate. In case of stable condition in the environment, the trend of sensor reading is fairly constant and predictable, therefore sensor node decreases its sampling rate. However, in case of unstable conditions which sensor readings changes more frequently, sensor nodes employ higher sampling rates to cover these frequent changes.

In this paper, we propose a new adaptive sampling algorithm which improves the performance of algorithm proposed in [7] in terms of energy consumption and data quality.

The rest of this paper is structured as follows: Section II presents related works while Section III explains our system and energy models. Data correlation concepts which are utilized in adaptive sampling approaches are described in Section IV, while our adaptive sampling approach is described in Section V. In Section VI, performance evaluation and simulation are presented. Finally, Section VII provides the conclusion.

II. RELATED WORKS

Many existing works utilize temporal, spatial, or spatio-temporal correlations to adapt sampling frequency. Authors of [8] present a centralized adaptive sampling technique, which utilizes temporal correlation. Padhy et al. consider temporal correlation while defining a utility-based sensing and communication protocol [11]. They model temporal correlations as a piecewise linear function and use a predefined confidence threshold to find an appropriate sampling frequency. Each node uses a linear regression model for its prediction. Use of time series forecasting methods to predict sampling and transmission rate is reported in [7]. Using cross layer information, this method adapts to the topology changes. Willet et al. consider spatial correlation-based adaptive sampling and propose a two steps algorithm [9]. In the first step, called the preview step, only a subset of nodes are turned on and send data to the sink. In the second step, called refinement, the sink may activate additional sensors in those locations where the spatial correlation is low. There are also some adaptive sampling

978-0-7695-4843-2/12 $26.00 © 2012 IEEE
DOI 10.1109/UIC-ATC.2012.156

techniques based on spatio-temporal correlation. For instance, the adaptive sampling approach of Gedik et al. [10] elects cluster heads and assigns nodes to the clusters based on sensor readings similarity and the hop counts. Cluster construction algorithm is executed periodically based on energy level and change of data behaviour in individual nodes.

III. System and Energy Models

A. System model

We consider a network composed of N stationary sensor nodes and C cluster heads which are deployed over an area. The location of sensor nodes, cluster heads and the base station are fixed and are known a priori. All sensor nodes are homogeneous in terms of sensor they have. Cluster heads are more powerful than sensor nodes in terms of processing capability.

Each sensor node has a set of sampling rates which is denoted by $SRSet=\{sr_1, sr_2, ..., sr_k\}$. This set represents the possible sampling rates which can be taken by a sensor node to observe the given area during its lifetime. It must be mentioned that every sensor node is allowed to use only one sampling rate for each time interval. In this structure the number of sampling nodes can vary from one time interval to another. In fact, spatial correlation among sensor nodes' readings in different time intervals demands different combination of sampling nodes. A set of possible sampling nodes for each cluster (such as h) is defined as $CSet_h=\{Cs_{1h}, Cs_{2h}, ..., Cs_{ph}\}$ where p is the total number of possible combinations and h is the specific cluster which this set belongs to it. Each Cs_{jh} consists of a specific combination of sensor nodes, which can be considered as sampling nodes.

For spatial-temporal correlation (also addressed in this study), we need to deal with both $CSet_h$ and $SRSet$. Therefore, it is likely that in each time interval, different combination of sensor nodes which are defined in $CSet_h$ monitor the area with different sampling frequency. It is clear that $|Cs_{rh}|$ can be different with $|Cs_{sh}|$ when $r \neq s$ as the number of sampling nodes participating in one combination can vary. Each Cs_{jh} is allowed to use only the combination of sampling frequencies whose size is equal to the number of sampling nodes participate on that Cs_{jh}. For example, for three nodes, we only use a combination of three different sampling frequencies. Therefore, we define a new set which is called SR_CSet_h , to show possible combinations of available sampling frequencies for each Cs_{jh} as follows:

$$SR\_CSet_h=\{SR\_Cs_{1h}, SR\_Cs_{2h}, ..., SR\_Cs_{ph}\}$$

Each SR_Cs_{jh} in SR_CSet_h corresponds to each Cs_{jh} in $CSet_h$. Each SR_Cs_{jh} consists of different combinations of sampling rates, each of which can be employed as a possible combination of sampling rates for Cs_{jh}.

B. Energy model

There is a specific resource cost associated with every data sampling policy. The resource which is of our interest in this study is energy that is the scarcest resource in the WSNs. Energy consumption has direct relation with the number of sampling sensor nodes active in the sampling process and their sampling frequency. We use the energy model discussed in [3] to calculate the energy consumption.

To be able to calculate the total energy consumption, it is necessary to find out the energy consumption of each operation, which is performed by sensor nodes and cluster head. Energy consumed by each sensor node and cluster head relates to five basic energy consuming tasks, namely, processing, radio transmission and receiving, transient energy, sensor sensing, and sensor logging. In case of cluster head, the total number of data packets that each cluster head receives and processes is the sum of the data packets sent by the sampling nodes of that cluster. We assume that packet size and transmission distance are fixed for all sensor nodes and all sensors (except cluster head) are homogenous.

We ignore the energy consumption for switching between ON and OFF states (transient energy). The formal energy consumption model for sensor node and cluster head are defined as:

$$E_{node}= E_{sens\text{-}node}+ E_{pro\text{-}node}+ E_{logg\text{-}node}+ E_{rx\text{-}node}+ E_{tx\text{-}node} \quad (1)$$

$$E_{CH}=E_{pro\text{-}CH}+ E_{logg\text{-}CH}+ E_{rx\text{-}CH}+ E_{tx\text{-}CH} \quad (2)$$

where $E_{sens\text{-}node}$ is the required energy for sensing, $E_{tx\text{-}node}$ is the required energy for transmitting one packet, $E_{rx\text{-}node}$ is required energy to receive control messages from cluster head , $E_{logg\text{-}node}$, and $E_{pro\text{-}node}$ are required energy for logging and processing data at sensor node, respectively. In case of cluster head, $E_{rx\text{-}CH}$, $E_{tx\text{-}CH}$ are the required energy for receiving one packet at the cluster head, $E_{logg\text{-}CH}$ and $E_{pro\text{-}CH}$ are energy dissipation for logging and processing readings of the sensor nodes at cluster head. On the basis of [3], transmission and reception power utilized here are 0.0495W and 0.0288W, respectively. Sensing power is 0.015W while logging and processing power are set to 0.016W and 0.0165W.

C. Data quality metric

Data quality is one of the most important QoS parameters in this work since resource management strategies must be satisfied. To be able to correctly quantify this QoS parameter, we have to evaluate the prediction of unavailable samples; which are calculated on the basis of utilization of spatial, temporal or spatial-temporal data correlation models. We consider two different definitions by looking at the quality requirements of the applications for data quality.

In the first case, consider an application that is interested in tracking every change in data readings. To this end, we are required to compare predicted data with observed or real data. In this case, a function is necessary to find out the value of the estimated data for the positions and timestamps at which samples are unavailable. To find the accuracy level of these estimations, the following equation can be utilized:

$$Err(X)=|X-R|,$$

where X represents the estimated value, and R represents the real observation. In fact, this equation calculates the amount of error between estimation and real data. If this error is less than a defined error threshold (Error_Threshold), it means that the environmental condition cannot be covered by the current sampling rate, or number of sampling nodes.

Table I. Proposed conditions to evaluate data prediction accuracy

Real Data	Prediction	Prediction Result
$R < a \ \|\| \ R > b$	$a \leq X \leq b$	Not-Accurate
$R < a \ \|\| R > b$	$X < a \ \|\| \ X > b$	Accurate
$a \leq R \leq b$	$a \leq X \leq b$	Accurate
$a \leq R \leq b$	$X < a \|\| X > b$	Not-Accurate

In the second case, an application introduces *[a, b]* as the pre-defined range, and expects to detect data which is out of this range. To this end, in our approach, each sensor node evaluates its prediction by considering Table I. According to Table I, when a sensor node correctly predicts that data is either in *[a b]* or out of that range, its prediction is accurate otherwise the prediction is un-accurate.

IV. Adaptive Sampling Concept and Evaluations

Spatial and temporal correlation of sensor nodes' readings is a unique characteristic of sensor networks, which can be exploited to achieve better performance in adaptive sampling techniques [2]. To be able to leverage the benefit of data correlation, some models are necessary to have prediction about samples based on correlation that exist between sensor readings.

As mentioned before, we aim to best manage network resource consumption by using proper data sampling policies while taking into account data quality and network lifetime as QoS requirements. To do so, changing the number and combination of sampling nodes (i.e. CSeth members) and sampling frequencies (i.e. SRSet members) are utilized as two data sampling policies. Studying and comparing the impacts of CSeth and SRSet members on energy and data quality can help find the best sampling node and the best sampling frequency for different time intervals. Increasing number of sampling nodes (in case of spatial-correlation) or sampling frequency (in case of temporal correlation) brings about high data quality but has also high impact on energy consumption. To cope with this conflict of interests, the right trade-off between energy and data quality should be found.

A. Temporal adaptive sampling concept

The similarity degree between consecutive sensor readings on a single sensor node can vary on the basis of temporal variation of the physical phenomenon. This similarity is called temporal correlation, which can be employed to adjust sampling frequency of a sensor node to be energy efficient while keeping data quality at an acceptable level.

Predicting sensor readings is feasible through applying models which benefit from the temporal correlation among sensor readings. The Auto- Regressive Moving Average model (*ARMA*) [4] is one of these models to provide reliable predication of sensor readings which exhibit temporal correlation.

To find the best sampling frequency for sensor nodes in different time intervals, which best satisfies the trade-off between energy efficiency and data quality, we introduce A and E as n*m and n*1 matrices, in which n=|SRSet| and m is the number of time intervals.

$$A^i = \begin{bmatrix} err_{11} & err_{12} & \cdots & err_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ err_{n1} & err_{n2} & \cdots & err_{nm} \end{bmatrix} \quad E = \begin{bmatrix} eg_1 \\ eg_2 \\ \vdots \\ eg_n \end{bmatrix} \quad B^i = \begin{bmatrix} b_{11} & b^i_{22} & \cdots & b_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix}$$

Each entity err_{jt} in the matrix A represents the average error which is produced for each sensor node i in case of sampling with sr_j sampling frequency in time interval t. The term eg_j in E represents the energy that sensor node i consumes in case of sampling with sr_j sampling frequency.

In order to study the effects of each sampling rate in different time intervals on energy efficiency and data quality, we need to make sure that only one sampling rate is selected for each node in each time interval. To do so, a binary matrix B is defined, whose rows and columns correspond to sampling frequencies and time intervals, respectively. After sensor nodes calculate their energy and error using energy model and data quality metric proposed in Section B, we define a utility function, which makes sensor nodes capable of selecting the sampling rates in order to minimize energy consumption and ensure high data quality. The utility function defined in Equation (3) is subject to the following: (i) each sensor node can select only one sampling rate at each time interval, and (ii) err_{jt} must be lower than a predefined error threshold.

$$UF_{temporal} = \arg \min \sum_{j \in SRSet, i \in Cl_k} err^i{}_{jt} \times eg_j \times b^i_{jt} \qquad \forall t \in Time\_Interval \quad (3)$$

Subject to:

$$\sum_{j=1}^{|SRSet|} b^i_{jt} = 1 \qquad \forall i \in Cl_h, t \in Time\_Interval \quad (4)$$

$$err^i_{jt} <= Error\ Threshold \qquad \forall i \in Cl_h, t \in Time\_Interval, j \in SRSet \quad (5)$$

The problem we deal with here is a simple linear programing in general and task assignment problem in particular. Task assignment problem becomes an issue when it is required to define how the tasks must be mapped into a given network. In each time interval, we have a given monitoring task for each sensor node, which must be accomplished using one of available sampling rates. To do so, for each time interval, we search and find the best sampling rate for each node so that it minimizes energy consumption of the network and maximizes data quality.

B. Spatial adaptive sampling concept

Due to the dense distribution of sensor nodes in a cluster, whenever an event occurs in a region, sensor nodes which are located close to each other may have high a degree of similarity among their readings. This concept brings the spatial correlation property to mind. Due to data similarity in the space domain, just a small number of sensor nodes from each region may be required to do sampling and send data to the base station in order to provide accurate data with an acceptable error level.

We utilize Multi-variable Normal (MVN) distribution model [5] to model spatial correlation among sensed data. "*The multivariate normal distribution is often used to describe, at least approximately, any set of (possibly) correlated real-valued random variable each of which clusters around a mean value*"[5].

While utilizing the spatial correlation model, we introduce *A* and E as *p*m* and *p*1* matrices, in which $p=|CSet_h|$ and *m* is the number of time intervals. Each entity err_{jt} in the matrix *A* represents the average error for a specific combination of sampling nodes Cs_j in time interval *t*. The term eg_j in E represents the energy consumption in case of sampling with specific combination of sensor nodes and *B* is defined to make sure that only one member of $CSet_h$ is selected in each time interval.

After the cluster head calculates sensor nodes' energy and error using the proposed energy model and data quality metric, we define a utility function which makes cluster head capable of selecting the combination of sampling nodes in order to minimize energy-data quality trade-off subject to two system constraints. The first constraint states that only one combination of sampling nodes must be selected for each time interval, whereas the second constraint states that err_{jt} must be lower than a defined error threshold.

$$UF_{spatial} = \arg \min \left(err_{jt} \times eg_j \times b_{jt} \right) \quad \forall t \in Time\_Intrval, j \in CSet \qquad (6)$$

Subject to:

$$\sum_{j=1}^{|CSet|} b_{jt} = 1 \qquad \forall t \in Time\_Interval \qquad (7)$$

$$err_{jt} <= Error\ Threshold \qquad \forall j \in CSet, t \in Time\_Interval \qquad (8)$$

C. Temporal-spatial adaptive sampling concept

It is often preferable to combine two previously mentioned models together, which aims to leverage the benefit from each. By doing so, the optimal combination of sampling nodes and their proper sampling frequencies can be identified through utilization of both spatial and temporal correlations.

In spatial-temporal correlation model, we aim to find the proper combination of sampling nodes with the best combination of sampling rates in different time intervals by taking energy-data quality trade-off into account. Approaching this goal, we consider the following two steps:

Step 1: For each Cs_{jh} we find the best combination of sampling rates which can satisfy the energy-error trade-off. Thereafter, for each time interval, the best Cs_{jh} which can satisfy the energy-data quality trade-off will be found. Again, at the first step, *Aji* and E_{ji} are changed to *q*m* and *q*1* matrices, in which $q=|SR\_CSet_{ji}|$ and m is the number of time intervals. Each err_{xt} in the matrix A_{ji} represents the average error for Cs_{ji} in case of sampling with the *xth* frequency of SR_Cs_{ji} in time interval *t*. The term eg_x in E_{ji} represents the energy consumption that corresponds to *xth* sampling frequency of SR_Cs_{ji}. A binary matrix *B* is defined such that each entity of this matrix shows which specific combination of sampling rates from SR_Cs_{ji} is selected for that time interval.

For each Cs_{ji} and given error and energy, we define a utility function which makes cluster head/sensor nodes capable of selecting the combination of sampling rates from SR_Cs_{ji} so that it satisfies energy-error trade-off subject to some system constraints. The first constraint, states that only one combination of sampling nodes must be selected for each time interval whereas that second constraint states that err_{xt} must be lower than a defined error threshold.

$$UF^{ji}_{1} = \arg \min (err_{xt}^{ji} \times eg_{xt}^{ji} \times b_{xt}^{ji}) \qquad \forall t, x \in SR\_Cs_{ji} \qquad (9)$$

Subject to:

$$\sum_{x=1}^{|SR\_Cs_{ji}|} b_{xt}^{ji} = 1 \qquad \forall t \in Time\_Interval \qquad (10)$$

$$err_{xt}^{ji} <= Error\ Threshold \qquad \forall x \in SR\_Cs_{ji}, \forall t \in Time\_Interval \qquad (11)$$

Step 2: The proper Cs_{jj} in different time intervals which best satisfies energy-error trade-off needs to be formal. To this end, *A* and *E* are changed to *p*m* and *p*1* matrixes which $p=|CSet_i|$ and *m* is the number of time intervals. Each err_{xt} in matrix *A* represents the average error for each Cs_j in time interval *t*. The term eg_x in *E* represents the energy consumption in case of sampling with the sensor nodes listed in Cs_j.

After error and energy are calculated by cluster head/sensor nodes, we define a utility function which makes cluster head/sensor nodes capable of selecting the combination of sampling nodes that satisfies error-energy trade-off subject to two system constraints. The first constraint, states that only one combination of sampling nodes must be selected for each time interval whereas that second constraint states that err_{xt} must be lower than a defined error threshold.

$$UF_{spatio-temporal} = \arg \min \left(err_{jt} \times eg_j \times b_{jt} \right) \quad \forall t \in Time\_Interval, \quad j \in CSet \qquad (12)$$

Subject to:

$$\sum_{x=1}^{|CSet_i|} b_{xt} = 1 \qquad \forall t \in Time\_Interval \qquad (13)$$

$$err_{xt} <= Error\ Threshold \qquad \forall x \in CSet_i, t \in Time\_Interval \qquad (14)$$

V. A DECENTRALIZED TEMPORAL CORRELATION BASED ADAPTIVE SAMPLING TECHNIQUE

The adaptive sampling mechanism presented here has a decentralized sampling policy using which each sensor node decides about its own sampling rate. When a sensor node experiences stability in its environmental condition, it reduces its sampling frequency. By doing so, number of data transmissions between the sensor node and the cluster head will also be reduced. Both sensor nodes and the cluster heads employ the same prediction model, i.e., *ARMA* which well describes the given environmental attributes. Utilizing *ARMA*, every sensor node transfers data to the cluster head only when environmental condition is not stable and thereby its data prediction is not accurate enough.

To ensure data quality, our approach employs the second definition of data quality metric in section III. User introduces *[a b]* as a normal data range and any data out of this range, is supposed as an event. Then, user specifies the quality requirements as a percentage of accuracy to detect event (i.e. data which is out of this range). As long as sensor correctly detects data is in *[a b]* or out of it, amount of difference between real data and sensor node prediction is not important. Therefore, sensor node does not track any small change in data readings which results in less observations and more energy saving. In this case it is only important to detect that data is in the range or out of it.

After being deployed, sensor nodes receive from the cluster head the error threshold level that can be tolerated by

the application. After that, each sensor node initially gathers the first *r* consecutive readings and stores them in a buffer. Each sensor node also sends these readings to the cluster head which exploits this history data to predict future data for its cluster members. Having *r* last readings, the readings for the epoch *(r+1)* is not only acquired but also predicted. The sensor node utilizes *ARMA* model to predict $(r+1)^{th}$ reading and then compares it with its real reading. We refer to the n^{th} reading acquired after the *r* readings in the buffer as *R* and the n^{th} reading predicted after the r readings in the buffer as *X*.

If the prediction is accurate enough, sensor node assumes that predication approach can be utilized to the next epoch. This, as shown in table I, happens when sensor node prediction and real data are both in or out of *[a b]*. In this case, the prediction is accurate and sensor node is still able to detect changes. Therefore it skips the next reading (i.e. the $(r+2)^{th}$ reading) which can be predicted by *ARMA* model. To do so, LastSkipSample (*LSS*) is defined here and sets to 1. This variable indicates the number of readings which can be skipped before the next sampling. In order to track number of skipped sample, another variable SkipSample (*SS*) is defined and set to the value held by *LSS*. For each skipped sample, the value of *SS* is decreased by 1. Once SS reaches zero, sensor node samples another reading and compares it with its prediction. If the prediction is accurate, *LSS* is increased by 1 and *SS* is set to the new value of *LSS*.

In case of having an accurate prediction (see Table I), sensor node does not send any data to the cluster head because the cluster head utilizes the same data prediction model and therefore it knows the value. Thus in this case, *SS* reduces to 0 in epoch *(r + 3)* and a sensor reading, *R* is acquired and prediction, *X* is calculated. This time, if *X and R are* both in or out of *[a b]*, *LSS* is incremented by one (i.e. it is set to 2) and *SS* is then set to the value held by *LSS*. In short, for each accurate prediction *LSS* increases by 1 and *SS* decreases to zero from an initial starting value of *LSS*.

This process continues as long as predictions are accurate. When predictions become inaccurate, sensor node has to sample at each time stamp. To do so, *LSS* and *SS* are set to zero and sensor node senses till accurate prediction obtained. Also, in case of inaccurate prediction, sensor node sends its reading to the cluster head in order to update the prediction model of the cluster head. Figure 1 illustrates the procedure.

It is worth mentioning that in case of having a long line of correct predictions, *LSS* is increased infinitely which may lead to miss some changes happened in the measured parameter. To cope with it, MaximumSkipSamplesLimit (*MSSL)* is defined to limit maximum number of samples which can be skipped. Once LSS reaches to MSSL, it remains unchanged even if further correct predictions are made. Maximum value which can be assigned to MSSL is 2 less than the buffer size [7].

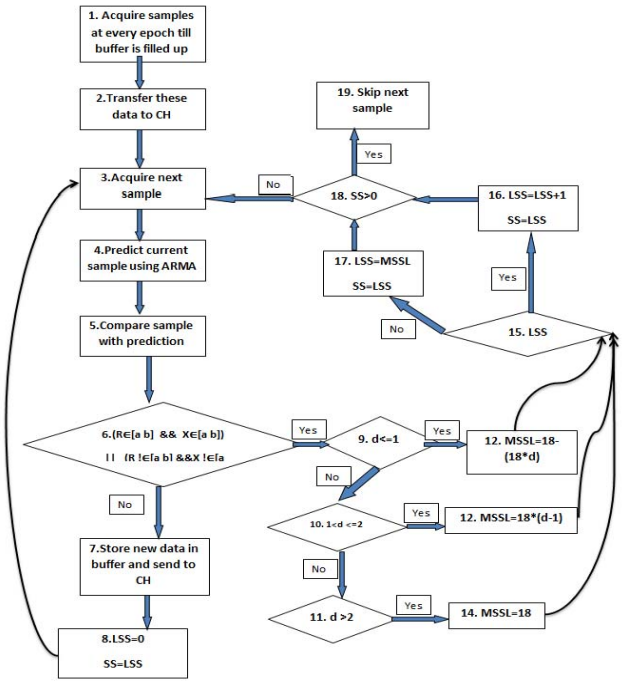

Figure. 1. Flowchart of our decentralized temporal correlation based adaptive sampling algorithm

Till here, our approach has some similarities with approach of Chatterjea and Havinga [7]. However, we now improve their technique by employing a new solution to adjust *MSSL* to the environmental condition and sensor node readings. This solution makes sensor nodes capable to adjust their *MSSL* parameter autonomously which satisfy user requirements in terms of event detection accuracy. Besides, it can detect events with any duration.

In order to adjust *MSSL*, sensor node takes the amount of difference between its readings with the mean of data range into account. Regarding this difference, we define three different regions for sensor readings (Figure 2). Region A includes readings which are between mean and borders of *[a b]* while Region B and C are for the data which are out of the range. In this figure upper bound is a and lower bound is b.

As it can be seen from Figure 2, sensor node reading can be in any of three separate regions for each of which sensor node employs different formula to calculate *MSSL*. The amounts of difference between sensor node readings and mean of *[a b]* define these regions. However, sensor node sets the difference between border of range (a or b) and mean as the maximum difference (|*a-mean*| *or* |*a-mean*|) which can be between sensor reading and border. Then, this maximum difference is utilized to introduce a new coefficient *d* which describes the ratio of difference between sensor node readings to the maximum difference *(*|*a-mean*| *or* |*a-mean*|). This coefficient is used by maximum allowed value of MSSL $(Buff_{size} - 2)$ to adjust MSSL.

$$d = \frac{|data - mean|}{|a - mean|}$$

In following we elaborate on the solution which is suggested to adjust MSSL:

- In region A, *d* varies between zero and one (*0<=d<=1*) based on the reading distance to the mean. *MSSL* is adjusted as follows:

$$MSSL = \lceil (Buff_{size} - 2) \times (1 - d) \rceil$$

Therefore, when sensor node reading equals to mean (*d=0*), sensor node can skip maximum number of samples which is defined in [7] ($MSSL = Buff_{size} - 2$). In other hand, *a* and *b* are two critical points which sensor node cannot skip anymore (*d=1*), so for these points we have *MSSL=0*.

When sensor node reading is close to the mean, value of *d* is small and *MSSL* becomes bigger, then sensor node skips more samples while coming towards boundaries, the value of *d* increases which leads to smaller values for MSSL.

- In region B, readings change between *[a-mean]* and *2*[a-mean]* or between *[b-mean]* and *2*[b-m1n]* therefore *1<d<=2*. Regarding that, *MSSL* is updated using following formula:

$$MSSL = \lceil (Buff_{size} - 2) \times (d - 1) \rceil$$

Once sensor node reading is close to the boundary, *MSSL* takes small values while farther than boundary sensor node uses big MSSL values.

- In region C, we use fixed value for *MSSL*. Since sensor node reading are far from boundaries, *MSSL* can be assigned with maximum number of skips.

$$MSSL = Buff_{size} - 2$$

To summarize, for each accurate prediction, each sensor node considers the region of its readings and based on this, calculates the difference between its reading with the mean of *[a b]* to find coefficient *d*. Then it utilizes *d* as a coefficient of $(Buff_{size} - 2)$ to update *MSSL*. Hence, *MSSL* always changes between 0 to $Buff_{size} - 2$. Figure 2 illustrates the procedure.

VI. Performance Evaluation

We now utilize proposed data correlation models to analyze our adaptive sampling approach. Before analyzing simulation results, we need to introduce other scenarios which are used for comparison.

A. *Possible improvements*

Our algorithm changes some of the parameters in [7] to improve its performance. We analyze all possible combination of parameters and introduce 11 different scenarios (see Table II) to compare our approach with others. Each scenario defines one specific version of [7]. The most important parameters utilized in these 11 scenarios are described as follow:

- In [7], *LSS* indicates the number of readings which can be skipped before the next sampling. At each sampling point in time and for each accurate prediction, *LSS* increases by one till it reaches *MSSL*. In our simulations for some scenarios *LSS* is removed from the algorithm which means for every accurate prediction, *SS* has been assigned by *MSSL*. Therefore, sensor node directly skips

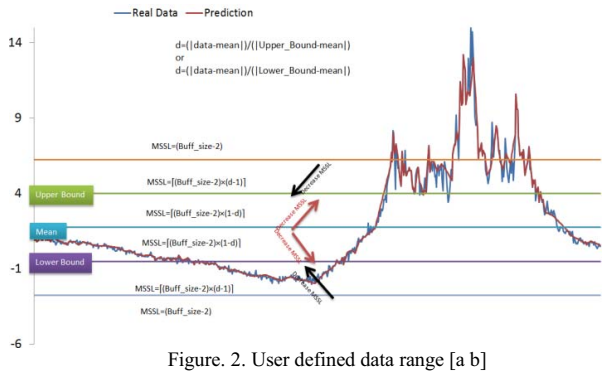

Figure. 2. User defined data range [a b]

MSSL number of samples for each accurate prediction. This change is shown in the column 3 of table II.

- In [7], sensor node has no prediction for skipped samples while it is possible to employ prediction model to estimate skipped samples values. To do so, for some scenarios presented in table II, sensor nodes predict value for skipped samples. These scenarios are mentioned in column 4.
- Adjusting *MSSL* to environmental condition is the most important change which is made in [7]. In column 2 of table II, three different formulas to adjust MSSL parameter, are suggested.

B. *Simulation results*

We use the given dataset [6] containing temperature readings collected for a period of two months. For simulation, we assume that data range is [-0.5 4], buffer size is 20 and user defines 90 percentage event detection accuracy as the data quality requirement. The network architecture which is utilized in this simulation is consists of several single-hop clusters. We consider only one cluster which consists of five sensor nodes. As shown in Figure 3, for the given dataset, sensor node experiences different environmental conditions. This figure illustrates real data and sensor node predictions. It can be seen from the Figure 3 that sometimes readings are in range and sometimes, it is out of the range. For the time intervals that data is close to the boundaries or data has fluctuation, sensor nodes require to skip less samples to be able to detect data which are out of the ranges while for the times that data are close to the mean of the range or farther than boundaries, it skips more samples. Observation points and produced error for our scenario have shown in Figures 4 and 5. As it can be seen, more samples are taken in the points which are close to the boundary while for farther points, sensor node experiences less observation.

Figure 6, shows that all scenarios detect events (whether data is out of the range) with at least 90 percentage accuracy. Therefore, all scenarios guarantee user defined accuracy requirement. However, those scenarios which update their *SS* parameter based on *LSS* show better accuracy levels. Scenarios such as scenario 1,3,7 and 9 utilize this policy and for each accurate prediction increase their *LSS* by one till it reaches to *MSSL*. Therefore, they can detect events in early stage and most important, their average error is less than others (Figure 7). In case of other scenarios such as scenario

2,4 and 11 which update their SS by MSSL, it is likely to have long skip which prevents them to detect events during that long skip as well as considering error level. Scenario 9 and our approach which employ the first policy propose the best accuracy level and average error among others since they completely tune MSSL values dynamically. It means that in each sample point, MSSL is adjusted based on its exact distance to the mean of data, while other scenarios partially or completely use fixed values for MSSL in different regions. Scenario 9 has better result rather than our approach because for each skipped sample, sensor node predict value for that epoch while in case of our algorithm, sensor node has no prediction for that epochs. As it can be seen in figure 7, proposed algorithm in [7] provides less average error rather than our approach since it tracks every small changes in sensor readings. However, the results are so similar, this shortcoming is negligible.

Furthermore, having higher accuracy and reduced average error result in additional cost in terms of energy consumption. Scenarios which consider first policy (*SS=LSS*) take more samples to provide higher accuracy levels. Sensor nodes updates their number of skips step by step till it reaches MSSL. Therefore, they consume much more energy than other scenarios which employ the second policy to update their SS. For the second policy, sensor nodes immediately skip MSSL number of samples which leads to less energy consumption. Our algorithm as well as other scenarios consumes less energy than [7]. Meanwhile, among most of other scenarios, energy usage of our algorithm is high since it does more calculation and is sensitive for any small difference between its reading and mean of data (Figure 8).

Considering these results, an evaluation metric is required to compare energy and error parameters together and make a trade-off between them. To do so, we use energy*error metric which was suggested in the section III. As it is described in Figure 9, most of the scenarios improve or provide same performance in comparison with previous adaptive sampling approach [7], but our policy is the best. Therefore this solution can be used as an improved mechanism which consume less energy while satisfying user defined accuracy levels.

I. CONCLUSION

We present a decentralized temporal correlation based adaptive sampling approach. The proposed mechanism has a decentralized sampling policy, using which, each sensor node decides about its own sampling rate. It introduces MaximumSkipSamplesLimit (MSSL) as a restriction for maximum number of samples which can be skipped by each sensor node. Our study proves that choosing an appropriate value for MSSL, that helps to detect events with high accuracy, is quite challenging. A small MSSL value increases the number of taken samples, while increasing event detection probability. On the other hand, taking a bigger MSSL into account is energy efficient but increases the probability of missing an event. Our proposed approach suggests a new dynamic way to adjust MSSL to the environment condition. In this approach, the user defines a data range which it expects sensor readings fall in, and asks network to detect data that is out of this range. In order to satisfy user defined quality requirements, we consider mean of this range, and update MSSL based on its distance to the mean. Closing to the mean, MSSL takes bigger values while coming towards boundaries, sensor node assign smaller values to MSSL. The approach allows sensor nodes to adjust their sampling frequency autonomously on the basis of environment conditions. Comprehensive experiments on the given data set indicate a reduction in sampling rates and data transmission, thereby leading to energy savings. On other hand, sensor nodes can detect events (data out of range) with a 90 percent accuracy level. Simulation results show that our approach provides better performance in terms of energy and data quality.

Table II. Different policies employed in improved adaptive sampling approach

	MSSL	SS	Prediction
Scenario 1	$\begin{cases} if\ d \le 0.4 & \rightarrow MSSL = 18 \\ if\ 0.4 < d \le 0.85 & \rightarrow MSSL = 18 - (18 * d) \\ if\ 0.85 < d \le 1.15 & \rightarrow MSSL = 1 \\ if\ 1.15 < d \le 2 & \rightarrow MSSL = 18 * (d-1) \\ if\ d \ge 2 & \rightarrow MSSL = 18 \end{cases}$	SS=LSS	
Scenario 2	$\begin{cases} if\ d \le 0.4 & \rightarrow MSSL = 18 \\ if\ 0.4 < d \le 0.85 & \rightarrow MSSL = 9 \\ if\ 0.85 < d \le 1.15 & \rightarrow MSSL = 1 \\ if\ 1.15 < d \le 2 & \rightarrow MSSL = 9 \\ if\ d \ge 2 & \rightarrow MSSL = 18 \end{cases}$	SS=MSSL	✓
Scenario 3	$\begin{cases} if\ d \le 0.4 & \rightarrow MSSL = 18 \\ if\ 0.4 < d \le 0.85 & \rightarrow MSSL = 9 \\ if\ 0.85 < d \le 1.15 & \rightarrow MSSL = 1 \\ if\ 1.15 < d \le 2 & \rightarrow MSSL = 9 \\ if\ d \ge 2 & \rightarrow MSSL = 18 \end{cases}$	SS=LSS	
Scenario 4	$\begin{cases} if\ d \le 0.4 & \rightarrow MSSL = 18 \\ if\ 0.4 < d \le 0.85 & \rightarrow MSSL = 18 - (18 * d) \\ if\ 0.85 < d \le 1.15 & \rightarrow MSSL = 1 \\ if\ 1.15 < d \le 2 & \rightarrow MSSL = 18 * (d-1) \\ if\ d \ge 2 & \rightarrow MSSL = 18 \end{cases}$	SS=MSSL	
Scenario 5	$\begin{cases} if\ d \le 1 & \rightarrow MSSL = 18 - (18 * d) \\ if\ 1 < d \le 2 & \rightarrow MSSL = 18 * (d-1) \\ if\ d > 2 & \rightarrow MSSL = 18 \end{cases}$	SS=MSSL	
Scenario 6	$\begin{cases} if\ d \le 0.4 & \rightarrow MSSL = 18 \\ if\ 0.4 < d \le 0.85 & \rightarrow MSSL = 9 \\ if\ 0.85 < d \le 1.15 & \rightarrow MSSL = 1 \\ if\ 1.15 < d \le 2 & \rightarrow MSSL = 9 \\ if\ d \ge 2 & \rightarrow MSSL = 18 \end{cases}$	SS=MSSL	
Scenario 7	$\begin{cases} if\ d \le 0.4 & \rightarrow MSSL = 18 \\ if\ 0.4 < d \le 0.85 & \rightarrow MSSL = 18 - (18 * d) \\ if\ 0.85 < d \le 1.15 & \rightarrow MSSL = 1 \\ if\ 1.15 < d \le 2 & \rightarrow MSSL = 18 * (d-1) \\ if\ d \ge 2 & \rightarrow MSSL = 18 \end{cases}$	SS=LSS	✓
Scenario 8	$\begin{cases} if\ d \le 1 & \rightarrow MSSL = 18 - (18 * d) \\ if\ 1 < d \le 2 & \rightarrow MSSL = 18 * (d-1) \\ if\ d > 2 & \rightarrow MSSL = 18 \end{cases}$	SS=LSS	✓
Scenario 9	$\begin{cases} if\ d \le 0.4 & \rightarrow MSSL = 18 \\ if\ 0.4 < d \le 0.85 & \rightarrow MSSL = 9 \\ if\ 0.85 < d \le 1.15 & \rightarrow MSSL = 1 \\ if\ 1.15 < d \le 2 & \rightarrow MSSL = 9 \\ if\ d \ge 2 & \rightarrow MSSL = 18 \end{cases}$	SS=LSS	✓
Scenario 10	$\begin{cases} if\ d \le 0.4 & \rightarrow MSSL = 18 \\ if\ 0.4 < d \le 0.85 & \rightarrow MSSL = 18 - (18 * d) \\ if\ 0.85 < d \le 1.15 & \rightarrow MSSL = 1 \\ if\ 1.15 < d \le 2 & \rightarrow MSSL = 18 * (d-1) \\ if\ d \ge 2 & \rightarrow MSSL = 18 \end{cases}$	SS=MSSL	✓
Scenario 11	$\begin{cases} if\ d \le 1 & \rightarrow MSSL = 18 - (18 * d) \\ if\ 1 < d \le 2 & \rightarrow MSSL = 18 * (d-1) \\ if\ d > 2 & \rightarrow MSSL = 18 \end{cases}$	SS=MSSL	✓
Improved Version	$\begin{cases} if\ d \le 1 & \rightarrow MSSL = 18 - (18 * d) \\ if\ 1 < d \le 2 & \rightarrow MSSL = 18 * (d-1) \\ if\ d > 2 & \rightarrow MSSL = 18 \end{cases}$	SS=LSS	

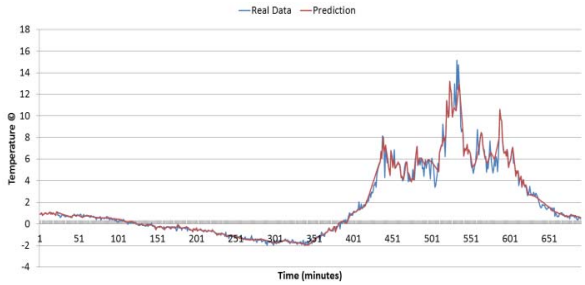

Figure. 3. Comparison between real data and prediction data

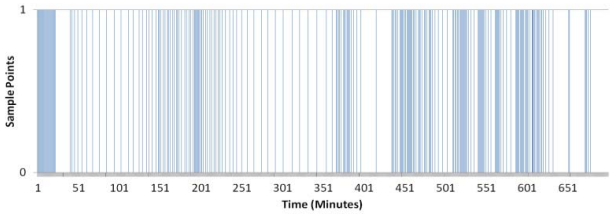

Figure. 4. Sample points

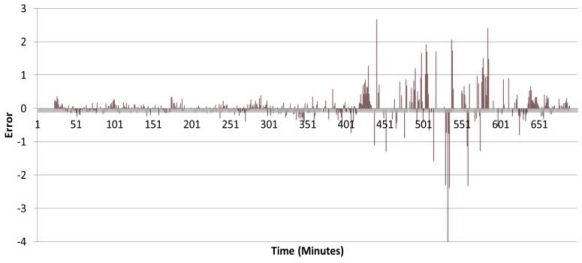

Figure. 5. Produced error during the time

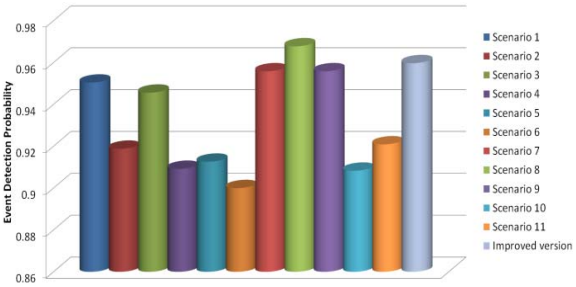

Figure. 6. Event detection probability for different policies

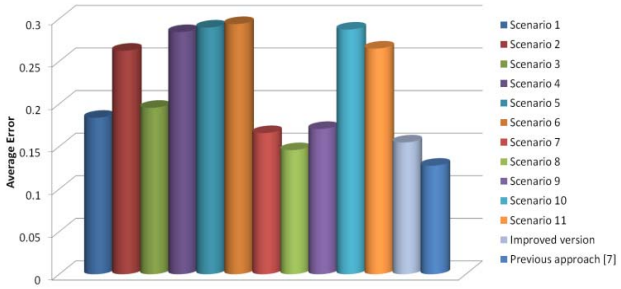

Figure. 7. Average error comparison among different policies

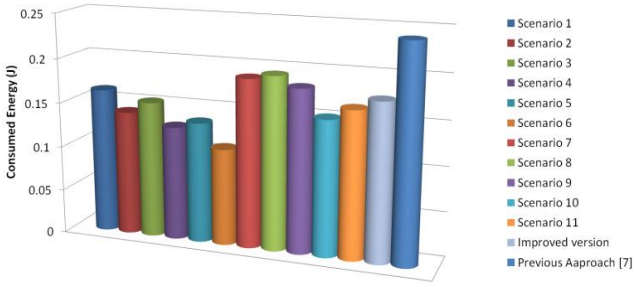

Figure. 8. Average energy consumption for different scenarios

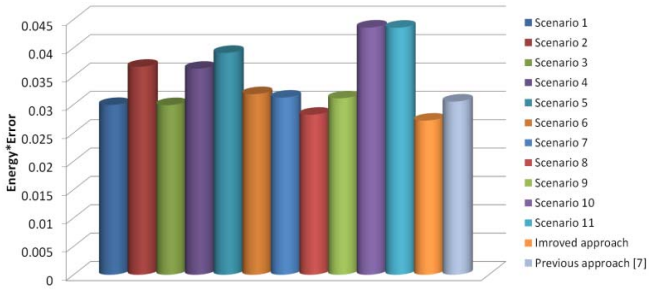

Figure. 9. Performance evaluation of different scenarios

REFERENCES

[1] G. Mainland G, D.C. Parkes, M. Welsh; "Decentralized, Adaptive Resource Allocation for Sensor Networks" Published in: • Proceeding NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2

[2] A. Deligiannakis, and Y. Kotidis; "Exploiting Spatio-temporal Correlations for Data Processing in Sensor Networks" Springer-Verlag Berlin Heidelberg , LNCS 4540, pp. 45–65, 2008.

[3] M.N. Halgamuge, M. Zukerman , K. Ramamohanarao, H.L. Vu, "An estimation of sensor energy consumption," Progress In Electromagnetics Research B, Vol. 12, 259-295, 2009.

[4] ARIMA ; http://en.wikipedia.org/wiki/Arima

[5] MVN;en.wikipedia.org/wiki/Multivariate_normal_distribution

[6] Aiello G, Scalia G.L , Micale R "Simulation analysis of cold chain performance based on time–temperature data", Production Planning & Control, DOI:10.1080/09537287.2011.564219

[7] S. Chatterjea, and P.J.M Havinga, "An Adaptive and Autonomous Sensor Sampling Frequency Control Scheme for Energy-Efficient Data Acquisition in Wireless Sensor Networks. In: 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), 11-14 Jun 2008, Santorini, Greece. pp. 60-78. Lecture Notes in Computer Science 5067. Springer Verlag. ISSN 0302-9743 ISBN 978-3-540-69169-3

[8] J. Zhou, D. De Roure,: "FloodNet: Coupling Adaptive Sampling with Energy Aware Routing in a Flood Warning System", Journal of Computer Science and Technology, Vol. 22, N. 1, pp. 121-130, January 2007

[9] R. Willett, A. Martin, R. Nowak, "Backcasting: Adaptive Sampling for Sensor Networks", Proc. International Symposium on Information Processing in Sensor Networks (IPSN 2004), pp. 124-133, 26-27 2004.

[10] B. Gedik, L. Liu, P. S. Yu, "ASAP: An Adaptive Sampling Approach to Data Collection in Sensor Networks", IEEE Trans. Parallel Distributed Systems, Vol. 18, N. 12, December 2007

[11] P. Padhy, R. K. Dash, K. Martinez, N. R. Jennings, "A Utility-Based Sensing and Communication Model for a Glacial Sensor Network", Proc. International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, 2006