# Collaborative Scheduling in Highly Dynamic Environments Using Error Inference

Qingquan Zhang*, Yu Gu*, Lin Gu‡, Qing Cao§ and Tian He*

*Department of Computer Science and Engineering, University of Minnesota Twin Cities
‡Department of Computer Science and Engineering, Hong Kong University of Science and Technology
§Department of Electrical Engineering and Computer Science, University of Tennessee Knoxville

*Abstract*—**Energy constraint is a critical hurdle hindering the practical deployment of long-term wireless sensor network applications. Turning off (i.e., duty cycling) sensors could reduce energy consumption, however at the cost of low sensing fidelity due to sensing gaps introduced. Existing techniques have studied how to collaboratively reduce the sensing gap in space and time, however none of them provides a rigorous approach to confine sensing error within desirable bounds. In this work, we propose a collaborative scheme called CIES, based on the novel concept of *error inference* between collaborative sensor pairs. Within a node, we use a sensing probability bound to control tolerable sensing error. Within a neighborhood, nodes can trigger additional sensing activities of other nodes when *inferred* sensing error has aggregately exceed the tolerance. We conducted simulations to investigate system performance using historical soil temperature data in Wisconsin-Minnesota area. The simulation results demonstrate that the system error is confined within the specified error tolerance bounds and that a maximum of 60 percent of the energy savings can be achieved, when the CIES is compared to several fixed probability sensing schemes such as eSense. We further validated the simulation and algorithms by constructing a lab test-bench to emulate actual environment monitoring applications. The results show that our approach is effective and efficient in tracking the dramatic temperature shift in highly dynamic environments.**

## I. Introduction

Wireless Sensor Networks (WSNs) have been used in many monitoring applications. Due to the small form factor and low cost of sensor nodes (e.g. the Mica series), they are normally equipped with limited power sources. If working continuously, a sensor node can typically sustain only a few days. On the other hand, long-term applications [1] are normally required to last for weeks or even months. The discrepancy between limited resources and stringent requirements makes it necessary to develop scheduling protocols to turn on and off (i.e., duty cycle) sensors to conserve energy.

Research on collaborative sensing is nothing new [2], [3], [4], [5], [6]. Most of projects focus on how to efficiently select or deploy a minimum set of sensor nodes to provide a full/partial spatiotemporal coverage. We note these work determines sensing activities of nodes based on coverage requirements in *space* and/or *time*. None of them focuses on how to schedule sensing activities based on *sensing error* and hence failed to provide a rigorous approach to confine data accuracy within desirable bounds.

In this work, we take a completely different approach. We schedule sensing activities based on two types of information: (i) local estimated error and (ii) inferred error from neighboring nodes. A node turns on its sensors when either error type exceeds a user specified tolerance. Our design has several major advantages over existing single-node scheduling methods [7]: (i) nodes can share sensing error information and process them with limited resources, (ii) nodes can collectively control sensing errors through neighborhood coordination, and (iii) a network can respond to dramatic environmental changes more quickly, a property that is desirable in environment monitoring applications.

Specifically, our design exploits neighbor node coordination to reduce possible violations against sensing fidelity requirements. The driving idea of our work is *error inference*, where the term *error* is defined as the difference, in percentage, between the ground truth environmental data and the corresponding value generated by the predictor of sensor nodes, which is a direct performance indicator of the sensor system. Not only is the error information used by the local sensing scheduler, but it is also shared among neighbors. Nodes can trigger additional sensing activities of other neighboring nodes when the inferred error has aggregately exceed the tolerance. We refer to our proposed approach as *Collaborative Inferred Error Sensing (CIES).*

The main contributions of this work are:

- The design of a local error control algorithm to guarantee a specified error bound;
- The introduction of a distributed inference model of prediction error for neighboring sensors
- The integration of both local and neighbor error control into a unified architecture to adjust duty cycles of sensor nodes.
- The simulated study and test-bed implementation of the proposed design that conserves as much as 60% of energy compared to other solutions, while confining sensing error within specified error tolerance.

The rest of this paper is organized as follows. Section VI describes the motivation behind our work from an application perspective. We present the overview for our design in Section II. Section III and IV describe the details of the error control mechanisms. The performance evaluation is presented in Section V. Section VII concludes the paper.

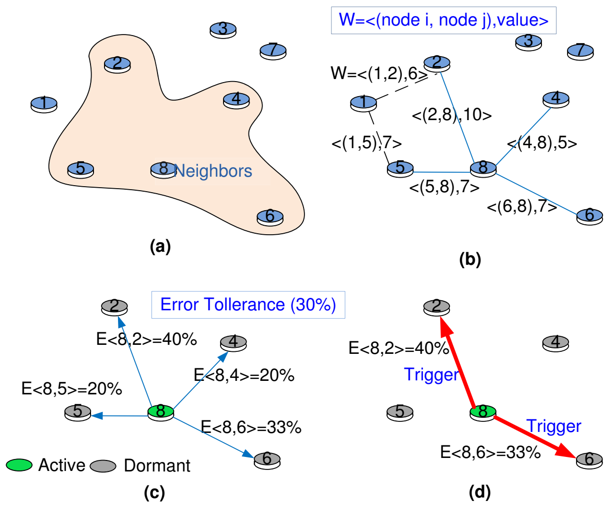

Fig. 1. Initial phase of the design

## II. Overview and Objectives

This section presents an overview of our Collaborative Inferred Error System (*CIES*). We first present the network model and assumptions of the work, then describe the overall system design.

### A. Network Model

Assume a wireless sensor network is composed of $N$ sensor nodes. Each sensor node has two states: an active state and a dormant state. An active node performs all the functionalities, such as sensing events, transmitting packets, and receiving packets. A dormant node turns off most functional modules except the radio for listening to incoming traffic. All nodes have their own schedules that are controlled by the duty cycle controller on the nodes. A dormant node wakes up when (i) it is scheduled to switch to active state, or (ii) it receives triggering packets from neighbors and decides to change into the active state.

### B. Assumptions

We assume that we use off-the-shelf sensor node products [8]. Without loss of generality, in our design and implementation, sensor nodes are homogeneous and can be distributed as a random process. We also assume that in our target sensing platforms, sensing is much more expensive than communication, so that it is necessary to coordinate sensing activities among neighboring nodes for better energy efficiency. Certainly, this assumption does not hold for all platforms, but it does apply to a few existing ones. For example, the magnetometers used in the MICA sensor boards and XSM nodes draw about 90 mA of current, as compared to 6mA for the ATmega128L micro-controller and 12 mA for the transceiver [9]. This assumption also holds well in platforms where expensive sensors (e.g. camera and micro-power-impulse radar (MIR)) and low-power-listening techniques [10] are used.

### C. A Walk-through of the Basic Operating Procedures

In this section, we overview the collaborative scheme of our design using a walk-through example. The key concept in *CIES* is to exploit the neighbors' resource to infer the conditions of inactive/sleep sensor nodes. A brief description as illustrated in Figure 1 presents the collaborative error control process.

**Figure 1(a), Stage I:** Each sensor node executes its local error control procedure (*IES*) independently. Meanwhile, the network error control starts to build the neighbor library using a neighborhood discovery service [11], [12]. In our example, node 8 detects node 1 to 6 as its neighbors. The local error control will handle local node error control and duty cycle management.

**Figure 1(b), Stage II:** The network error control generates *node-pair weighted graphs* to represent the correlation among sensors. Figure 1(b) lists the edge weight $w(i, j)$ of several nodes. The higher a weight is, the stronger correlation these two sensor nodes will have. Previous literature has pointed out that it is reasonable to assume that this correlation is relatively stable for a period of time [13].

**Figure 1(c), Stage III:** Sensor nodes are controlled not only by the local error control, but also are sharing their resource to assist the neighbors, a process enabled through the utilization of the neighbors sensing error inference. For example, a sensor node (e.g. node 8 in this case) is active in a certain cycle, and detects an unexplained difference between the real sensing data and the output of prediction model. Then it calls the *network error control* to generate inferred error of adjacent nodes using the weighted graph shown in Figure 1(c).

**Figure 1(d), Stage IV:** If the inferred reading of any neighbor node is larger than a specified bound (e.g. the tolerance threshold of errors), the network error control will send out a trigger message to those nodes. In this example, because the error tolerance configured by the system is $30\%$, node 8 will send trigger messages to nodes 2 and 6 . After a sensor node receives such a message, it will process and analyze the information in the network control. After then, the node can either remain turned off if the inferred error from network error control is not large enough to trigger the duty cycle control process, or be switched on if otherwise.

As illustrated by this walk-through, *CIES* exploits neighbors collaboration of sensor networks in event detection, thus showing a unique potential for reducing errors in environment monitoring applications. Compared to other local error control schemes [7], *CIES* extends the error control from local node level to the network level, setting the foundation for more complicated applications.

In the following, we describe the design in more detail. One key objective of this work is to develop a generic error control mechanism through which collaborative sensors can achieve error-bounded scheduling control in applications.

## III. Local Error Control

The design of the local error control is motivated by the observation that a sensor node should be able to run programs independently even in isolation. Therefore, the data detected

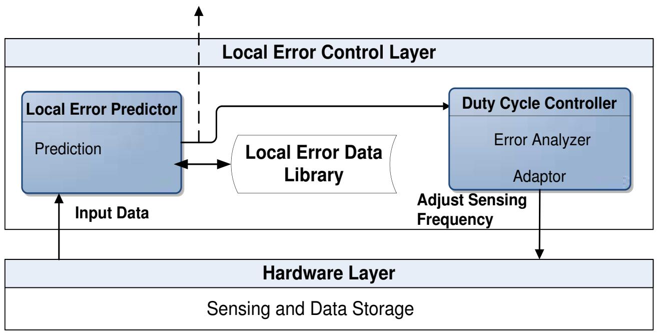

Fig. 2. The local error control layer illustration

Fig. 3. The preliminary temperature measurement experiments

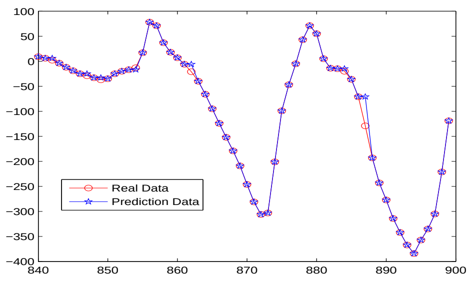

Fig. 4. An example of output from predictor (error tolerance 10%, x-axis is the time-stamp)

and stored locally should also be fault-tolerant, a goal that is achieved by the local error control

For convenience, we refer to the local error control as *Non-collaborative IES*, which supports routine applications that include the duty cycle control and local error predictor as shown in Figure 2. A sensor uses its local error predictor to predict the environment status without performing actual sensing operation. When data is obtained through actual sensing, a node compares predicted sensing values with the actual sensing values, and then store the prediction errors into the local error data library. Based on the accuracy of the local error predictors, the duty cycle controller adjusts the sensing frequency through *error bound control*, which serves to confine the system prediction error within a user specified bound.

## A. Local Error Predictor

The basic mechanism of local error predictor works as follows. In order to conserve their limited power supply, sensors do not continuously sense data. Instead, they operate at some selective cycles as long as the data quality is acceptable. The data in the remaining cycles are reconstructed through appropriate prediction models. If the environment exhibits cyclic patterns, an empirical model will be used to establish strong correlations in the data and to organize them in a certain way so that future data can be extracted from the empirical or historical ones.

Depending on system lifetime the data fidelity requirement of an application, the empirical model can be constructed in different ways. Similar to [14], we developed a cycle-based empirical model [15], which has been proven to be efficient for environment monitoring applications. The error predictor is mainly responsible for generating prediction errors, defined as $e_i$, for each node $i$. Our preliminary experiments of temperature measurements, as shown in Figure 3 and Figure 4, demonstrate that the error predictor can adapt to the environmental changes sufficiently well.

Since the energy resource on individual sensor nodes is limited, empirical model in this application domain can simplify data processing, and thus extend the lifetime of sensor nodes. However, the model selection can be flexible, and the *duty cycle controller* can adapt the system to the relative error induced by different prediction models. Moreover, the local error predictor provides a reliable reference for duty cycle controller to perform further analysis.

## B. Duty Cycle Controller

The duty cycle controller receives and analyzes the prediction errors from the local error predictor. The first step in designing the duty cycle controller is modeling sensing behavior of the system mathematically to derive the relationships among the local prediction error, the current duty cycle, and system requirements. In this design, we separate the controller into the error analyzer and duty cycle adaptor, two processes that can run collaboratively.

*1) Error Analyzer:* We determine the error analyzer theoretically as follows. We assume that the sensing baseline consists of $N$ data cycles, in which $k$ warm-up cycles will be used for building controller models. In each cycle, the probability that a sensor node performs actual sensing operation is defined as sensing probability $p_i$. To simplify the description, without loss of generality, instead of considering energy cost spent on different components, (e.g., sensing and processing), we use average energy consumption to represent the total energy cost of a node to sense, process and communicate in each sensing period. Let the average energy consumption for sensing be $E_a$. When a sensor is inactive, it does not sample the environment, instead it uses the local predictor to estimate sensing readings, which introduces prediction errors. Let the potential prediction error at each cycle be $e_i$. And let the maximum prediction error tolerance specified by a user be $e_t$ . Therefore, the goal of our design is to minimize the energy consumption during each baseline period:

$$E = \sum_{i=1}^{k} E_a \cdot t_i + \sum_{i=1}^{N-k} p_i \cdot E_a \cdot t_i \tag{1}$$

under the constraint that

$$\frac{\sum_{i=1}^{N} (1 - p_i) \cdot e_i}{N} = \frac{\sum_{i=k}^{N} (1 - p_i) \cdot e_i}{N} \leq e_t \tag{2}$$

where $t_i$ is the unit cycle length, $k$ is the length of cycles used to stabilize the scheduling system and $N$ represents the total length of operational cycles . The constraint will enforce that the potential statistical error caused by the prediction will be smaller than the error tolerance. The range of possible values of $p_i$ will be bounded to satisfy the constraint equation.

The minimization of energy consumption deals with several key issues, e.g. the length of the training cycle and the prediction model used. Now the problem is to determine the appropriate $p_i$ for a given error range $e_i$ obtained from past data values. To solve for sensing probability $p_i$ at a specific $e_i$ requires a joint distribution of a process for $e_i$ at specific time instance or period. This would require a heavy computation and storage overhead on the limited resource of the sensor node. Obtaining a solution for sensing probability $p_i$ will be extremely difficult to calculate during transitions. Instead, we introduce a lightweight method for computation that allows the sensor to choose the value within a range. We first determine the bound for sensing probability $p_i$, and the algorithm will choose one value within that bound.

It should be clear that the higher the value of sensing probability $p_i$, the larger the expected energy consumption. The lower the value of sensing probability $p_i$, the higher the probability for the error because that prediction will be greater than the tolerance. Therefore, we need to analyze the bound of sensing probability $p_i$ to optimize this trade-off.

*2) Determining the Sensing Probability Bound:* We use a bottom-up approach to set a bound for the sensing probability. That is, if we do not violate the error constraint in every cycle instance, we are certain that the inequality holds. As noted, this sets a stricter requirement than the constraint equation over all sampling instances. By doing so, our probability constraint problem can be simplified into choosing the $p_i$ at each scheduling cycle to satisfy the constraint on $(1 - p_i) \cdot e_i$, which can be solved as

$$p_i^{lb} = \begin{cases} 0 & 0 \leq e_i \leq e_t \\ \\ 1 - \frac{e_t}{e_i} & e_t \leq e_i \leq 1 \end{cases} \tag{3}$$

The $p_i^{lb}$ is the lower bound of $p_i$ that guarantees data quality requirement at each sensing cycle instance. Only values higher than this will assure that the constraint requirement won't be violated under any circumstances. We should also be careful in the selection of $p_i$, as a higher $p_i$ implies more energy consumption by the sensor node.

We also note in Equation 3 that the lower bound $p_i^{lb}$ is affected by the prediction error $e_i$. A large prediction error $e_i$ imposes a higher bound, leading to high energy consumption.

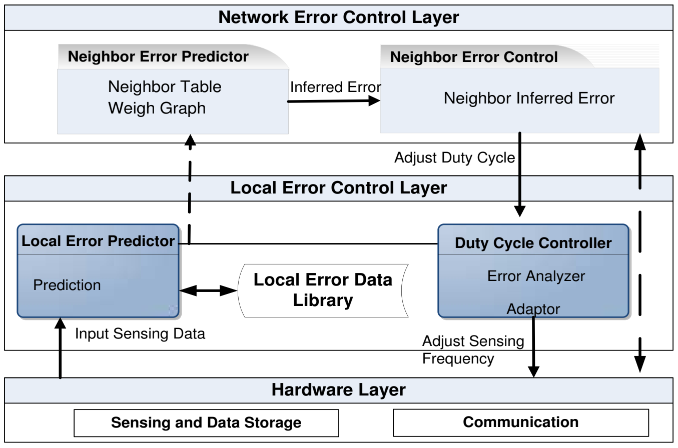

Fig. 5. The network error control layer illustration

The critical issue is to reduce this prediction error $e_i$. Clearly it can be achieved with a better prediction model, however when environment changes quickly and unpredictably, a prediction model based on historical data is not effective. Therefore we need online method to improve error control, which is achieved through the *Network Error Control* described in next section.

## IV. Network Error Control

In this section, we present the design of network error control as shown in Figure 5. Recall in our analysis in section III-B2, it is essential to predict the neighbors' model prediction error accurately and share such information among them effectively. To ensure the accuracy of such prediction and information sharing, we assign tasks to two processes: 1) neighbor error predictor, 2) neighbor error controller. The process running on this network control layer is named *collaborative IES*, which aims to maximize the energy saving and minimize the prediction error of sensor system.

Before further discussion, we define several terms used to describe the processes.

**Definition 1 (Inferred Error $e_{ij}$):** Given a node $i$ and its neighboring node $j$, the node-pair inferred error $e_{ij}$ is defined as the inference error at neighbor $j$ from the point view of node $i$.

**Definition 2 (Node-pair Weight $w_{ij}$):** The weight is defined as the extent of a node-pair's data correlation and indicates how similar the sensing observation is between two neighboring sensor nodes $i$ and $j$.

**Definition 3 (Error Probability Density Function $\rho(e)$ ):** The error PDF is a collection of distributions of detection errors in which the past detection errors for sensors are stored and processed so that the detection errors can be directly linked to corresponding occurrence probability. The neighboring nodes will exchange the error *PDF* locally. We can easily derive its statistical accrual error probability mass function( *PMF*) once the *PDF* is given.

### *A. Design of Neighbor Error Predictor*

Since the neighbors can change dynamically, we need to iteratively estimate the neighbors' prediction error, given a certain relationship among neighbors.

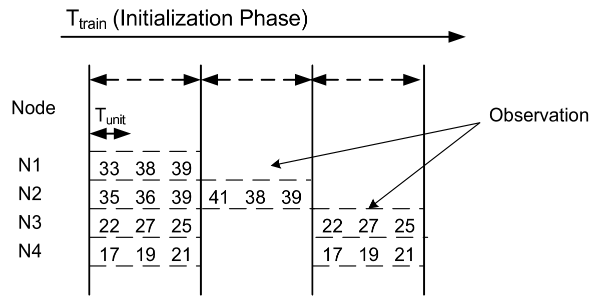

Fig. 6. The process of Correlation Calculation

**Step 1: Neighbor Recognition**

The control process starts with neighborhood discovery. During this phase, the sensors will acquire the knowledge that which close sensors around them can build up a “trust” relationship, which can be characterized as node-pair weights. The formation of neighborhood may be based on different requirements such as vicinity, link quality or the displacement along the routing path of the sensing data. In this stage, each node recognizes its neighboring nodes and assigns a table for each neighbor to build the weight graph. Note that the neighborhood formation is a dynamic stage which will be refreshed after a defined period. By the end of this sub-process, sensors will recognize their neighbors and data storage structures created for neighbors will also be initialized.

**Step 2: Weight Graph Construction** As pointed in our earlier assumptions, nodes are synchronized with each other, and $T_{train}$, the time for initialization, is divided into equal time durations $T_{build}$, as in Figure 6. Each time duration includes $m$ equal duration intervals, where an interval is a unit sample time period set by the user.

For each round, each node $N_i$ stores its observation vector $\{o_1^i, o_2^i, ..., o_m^i\}$ obtained through discrete sampling at $T_i = \{t_1^i, t_2^i, ..., t_m^i\}$. At the end of each round, each node exchanges the observation vector, which is used to calculate the correlation between nodes. This process is repeated until the end of $T_{train}$, so that the average sensing correlation between nodes can be estimated.

Specifically, we use the following approach to calculate data correlation between two observation vectors $C(i,j)$ by node $N_i$ and node $N_j$.

$$C(i,j) = \frac{m\sum o_k^i o_k^j - \sum o_k^i \sum o_k^j}{\sqrt{m\sum (o_k^i)^2 - (\sum o_k^i)^2}\sqrt{m\sum (o_k^j)^2 - (\sum o_k^j)^2}} \quad (4)$$

After getting the data correlation $C(i,j)$, we derive $w(i,j)$ as $abs(C(i,j) \times R)$ in which $R$ is a unified constant.

The weight value among nodes, once evaluated, remains constant throughout each short operation period, and is subjected to subsequent modifications to guarantees the accuracy.

The sensors propagate the information to neighboring sensors while receiving similar information, forming a graph through a one-to-one linear mapping. Eventually, based on the outcome from this sub-process, each sensor obtains essential node-pair weights which set the foundation to differentiate the status information in the error estimation.

*1) Costs and Complexity on Weight Graph Construction:* Here we will study the extra computation cost used for weight graph construction. With a maximum of $N$ sensor nodes within a group, there are a maximum of $N-1$ neighbors for each node. The total weight edges between two nodes will be

$$C_N^2 = \frac{N(N-1)}{2}$$

However, this value can be reduced after neighbor groups are formed because the total number of weight vectors will decrease as the nodes out of one-hop communication range are excluded. This can be expressed as: $C_N^2 - C_m^2 = \frac{N(N-1)}{2} - \frac{m(m-1)}{2}$ , in which $m$ is the number of nodes that are not in one-hop range. Therefore the computation costs for weight graph will be $O(N^2)$.

**Step 3: Achieving the inferred error $e_{ij}$ for neighbors**

The control of sensing errors in the network is further guaranteed by the collaboration of neighboring nodes. The observations that sensor nodes demonstrate spatial correlations found in [16], [17] are also supported by our preliminary experiments described in Section III. Motivated by such observations, we can predict error of neighboring nodes using local prediction error. That is, an active sensing node, by comparing its real-time sensing values with corresponding predicted values, can infer the prediction errors of correlated neighboring nodes. In this way, our neighbor-error inference scheme ensures real-time tracking and quick response to the error status change within a sensing group. The process can be summarized as follows:

- At a sampling cycle $m$, assuming sensor $i$ is active in sensing and computation, we can easily calculate the observation error $e_i^m$ at source node $i$ based on the difference between actual sensing data and prediction values that are generated by our prediction model.
- From its error *Probability Density Function* $\rho(x)$, node $i$ evaluates the cumulative distribution function *PMF*$_i(e_i^m)$ as in Equation 5. From this result, we can infer the statistical confidence level of the worst error occurrence at node $i$ as.

$$PMF_i\left(e_i^m\right) = \int_{-e_i^m}^{e_i^m} \rho\left(x\right) dx \quad (5)$$

- Upon obtaining the $PMF_i$, the active sensor node $i$ calculates the inferred error of a neighboring sensor $j$, based on *PDF* information of sensor node $j$. Given the neighbors error models, an iterative step is performed:

$$e_{ij} = PMF_j^{-1}\left(PMF_i\left(t\left[k\right]\right)\right) \quad (6)$$

and the variable $t\left[k\right]$ is expressed as:

$$t\left[k\right] = \begin{cases} 2 \cdot t\left[k-1\right] & PMF_j\left(t\left[k-1\right]\right) < PMF_i\left(e_i^m\right) \\ \frac{t[k-1]+t[k-2]}{2} & PMF_j\left(t\left[k-1\right]\right) > PMF_i\left(e_i^m\right) \end{cases} \quad (7)$$

in which $PMF^{-1}()$ is the inverse function of $PMF$ and $t\left[0\right] = 0, t\left[1\right] = e_i^m$. The iterative process will not stop until $PMF_j\left(t\left[k-1\right]\right) = PMF_i\left(e_i^m\right)$.

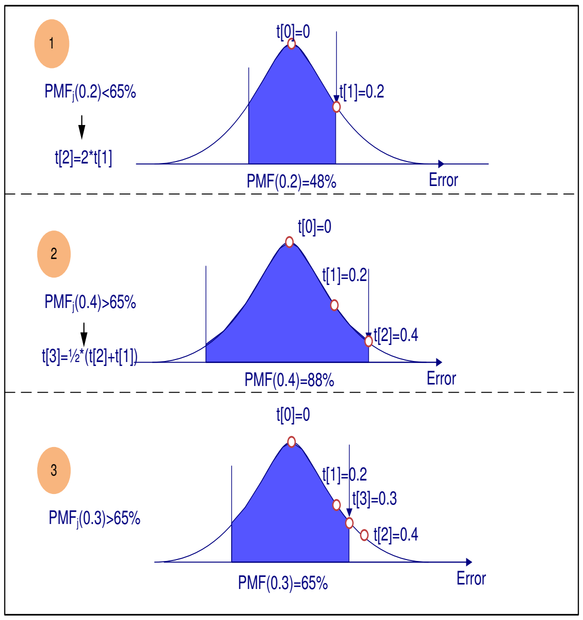

Fig. 7. The iterative inferred error computation

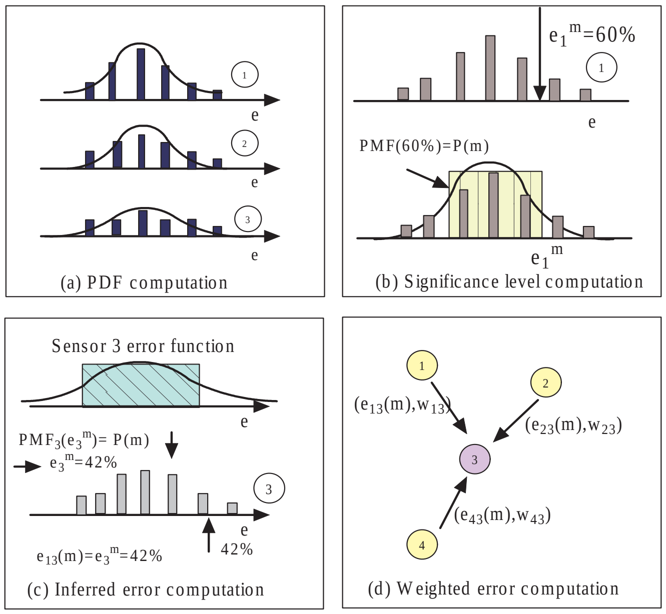

Fig. 8. The inferred error computation

Figure 7 shows an example of the inferred error computation process where neighbor control layer computes $PMF_j$ for node $j$ by using its own PMF value $PMF_i = 65\%$. The $t[1] = 0.2$ will be used as the first element in the iterative sequence. At step 1, the implied $PMF_j(t[1])$ becomes 48%, which is smaller than $PMF_i$. Then the computation point $t[2]$ will be $2 \times 0.2 = 0.4$. However, at step 2, the $PMF_j(t[2])$ is 88%, a number that is greater than 65%. Therefore, the next iterative point $t[3]$ will become $1/2 \times (0.2 + 0.4) = 0.3$. After step 3, the calculated $PMF_j(0.3) = 65\%$ and is the same as the $PMF_i$. The inferred error $e_{ij}$ is found and the computation process stops.

Because the different sensing characteristics between sensor pair have been incorporated into the weight value, then we can assume sensor $i$ and sensor $j$ share the same confidence level within the same sampling cycle. To further illustrate the above process, Figure 8 shows a simple example on error inference among neighboring nodes. In the figure, nodes 1 and 3 are neighbors which are closely related. At a given sampling time instance $m$, node 1 is active and starts sensing while sensor 3 is in dormant state. Based on the real sensing data and its predicted values from the predication model, sensor 1 decides the current detection error is 60%. Given this value, by looking at the *PDF* of sensor 1, we can know the desired accumulated probability or confidence level of sensor 1 is 0.58. Mapping this value onto sensor 3's error *PDF*, sensor 1 hence can estimate the inferred error of sensor 3 is 42%. Then if this 42% is larger than the error threshold, sensor 1 will send a wake-up message with this error information to sensor 3. Otherwise, there is no further actions performed by sensor 1.

### B. Neighbor Error Control

After estimating the error value for neighboring sensor $j$, sensor $i$ sends the inferred error $e_{ij}$ to neighbor error control process where those error information is sent and received. The neighbor error control process monitors the channel through which its neighbors send error information. To minimize the false positive risk, a weighted average approach is adopted in this design.

**Definition 4 (weighted average inferred error):** Given a neighborhood $G(V, E)$, a sensor $j$'s weighted average inferred error $e_j$ is the weighted average of all node-pair inferred errors, i.e, $e_{dj}$, where $d$ and $j$ are a neighborhood pair.

The weighted average error $e_j$ is obtained according to the following rationale. Node-pairs provide different inferred errors due to correlation relationships or other influences. Pair inferred errors have a specific weight value based on their degree of similarity. A higher weight value means a higher probability for data similarity. Therefore, the sensor platform must take this into account when determining whether the sensor needs to adjust its current operating status. Based on our observation, the inferred error can be expressed as:

$$e_j = \sum_k e_{kj} \cdot w_{kj} \Big/ \sum w_{kj}, k \in N(j) \qquad (8)$$

where $k$ is size of the neighborhood of the node $j$ , $w$ is the node-pair weight and $N(j)$ is the neighborhood list of node $j$. As shown in Figure 8 (d), sensor node $j$ receives several isolated error estimations from neighboring sensors $i$, $l$ and $q$. The $e_j$ should be viewed as a total effective contribution from all the neighboring inferred errors on a weighted basis. Apparently if one sensor detects that the estimation error currently violates the error threshold, its neighboring nodes having a high weight value are expected to experience a high risk of violating the data accuracy as well.

## V. EVALUATION

To evaluate, we develop a simulation program that uses historical soil temperature data. The temperature data were

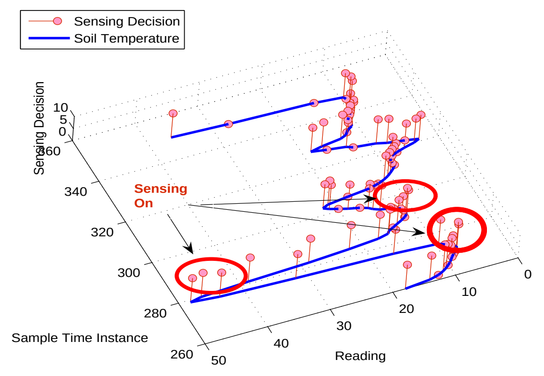

Fig. 9. The sample of sensor activities

collected from the Wisconsin-Minnesota Cooperative Extension Agricultural Weather Page [18] where soil temperature is monitored continuously, sampled twice per hour, 24 hours per day, for over 10 years. This data set is large enough to reduce sampling randomness, allowing us to verify our algorithm and investigate the impact of different configurations on the performance of energy conservation and error control. In our experiments, we use randomly deployed sensors in a square area of 200 m $\times$ 200 m. We define a pair of nodes as neighbors when their distance is less than 20 m. Moreover, we use a diffusive model to fill up the simulation environment, in which we consider the environment as a homogeneous semi-infinite medium. Various benchmark approaches such as the fixed probability sensing scheme are simulated to generate the metric data.

## A. Metrics and baseline

In order to evaluate the scheduling quality of a sensor network, we define metrics as follows:

- *Error Rate*. This metric is defined as the error rate that the prediction system produces during the same observation window.
- *Miss Ratio*: This metric is defined as the ratio that the sensor system fails to respond to an event, e.g., a rapid change in environment.
- *Energy Consumption*: This metric is defined as the total energy consumed by the network during the operation period.

The sensing schemes proposed will be assessed using the above metrics with respect to different system parameters, e.g. the error tolerance. Through these examples, comparison between different benchmarks and our proposed *CIES* will be used to demonstrate the performance of our design.

## B. The mechanism of our error bounded approach

One of the benefits of this adaptive and collaborative approach is that it will try to reduce the average error for the entire operation period. The stricter guarantee is that it will limit the error at each sampling instance, which will enhance the system performance. The difficulty in limiting the errors is to determine when to switch on the sensors whenever there is a dramatic change in the environment. Our approach achieves this by relying upon both local and network error control mechanisms. Local error control can guarantee the error bound when model-based prediction works well. However, when environments experience dramatic changes, model-based prediction no longer works. In such scenario, the network error control mechanism relies on active nodes to trigger inactive nodes, when the inferred error of the inactive nodes exceed the bounds.

As shown in Figure 9, when environment temperature experiences corner-like change, sensing activities becomes more intensive than other sampling periods. This is because that the network error control trigger more nodes to start sensing in order to avoid violation of the error bound.

## C. The detailed analysis of neighbor error control

In the first example, there are only two sensors adjacent to each other. This shows the performance of the simplest case of *CIES*.

Figure 10 illustrates the error performance of non-collaborative and collaborative $CIES$. Over error tolerances, we can see that both approaches can satisfy the error performance requirement. However, under the collaborative $CIES$ scheme, the error rate is at least 20% less than with stand-alone $IES$.

Figure 11 demonstrates the metric of miss ratio for stand-alone $IES$ and $CIES$. In the extreme region, (i.e., the error tolerance is 10%), the miss ratio is about 25% less with the collaborative information. The purpose is well demonstrated here.

Figure 12 demonstrates the energy consumption for both schemes. Compared to the 25% error rate improvement in $CIES$, the additional energy consumption is small as the maximum difference between the two schemes is only 5%.

From the above three figures, we can conclude that the $CIES$ method is superior to the stand-alone $IES$ scheme with slightly more energy consumption. This cost can be traded for the reduction in miss ratio, which has been considered important in certain monitoring applications [1].

## D. The impact of node density

In the second example, we raise the node density to 20 and study the effect of node density to performance metrics. Figure 13, 14, and 15 show the error rates associated with each approach. Compared to the results in Figure 10, the error rates are dramatically reduced while the gap between the two schemes is increased. This is expected because there is a greater chance for the sensor to be awakened by neighboring nodes.

From the miss ratio performance result, we can draw a similar conclusion. The miss ratio is reduced almost by 45% for different levels of error tolerance. This shows the validity of collaborative $IES$.

However, as shown in Figure 15, the energy consumption is slightly higher but still acceptable considering the improvement in miss ratio. It is the application's choice to balance the trade-off between energy consumption and other performance metrics.

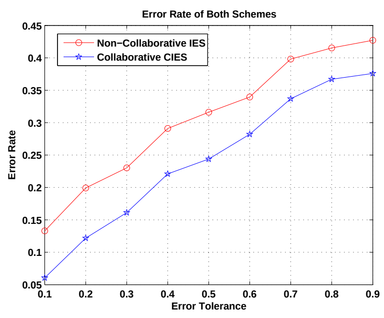

Fig. 10. The error performance with different error tolerance

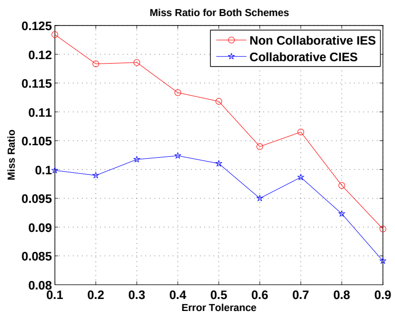

Fig. 11. The Miss Ratio with different error tolerance

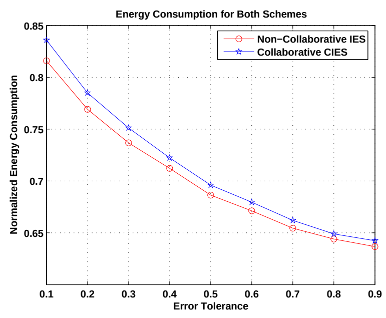

Fig. 12. The energy consumption with different error tolerance

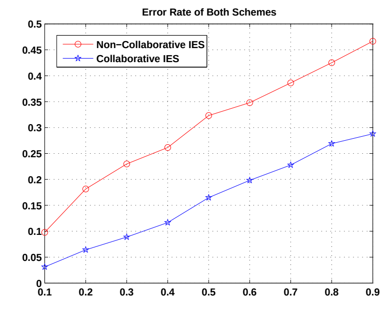

Fig. 13. The error performance with different error tolerance

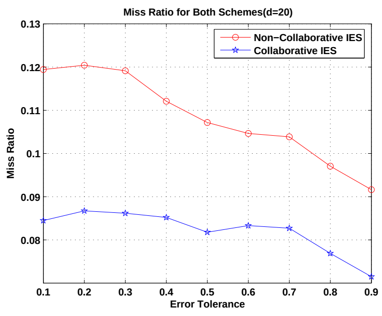

Fig. 14. The Miss Ratio with different error tolerance

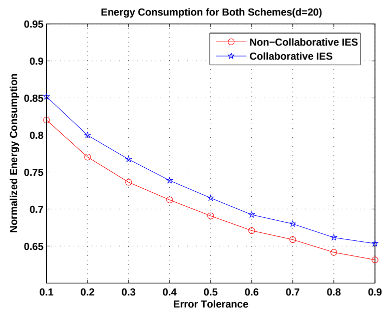

Fig. 15. The energy consumption with different error tolerance

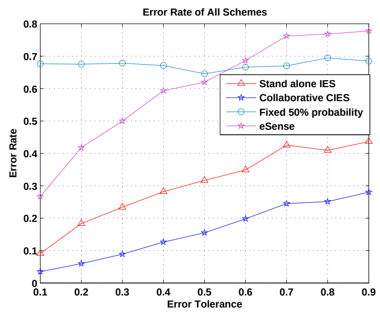

Fig. 16. The error performance with different error tolerance

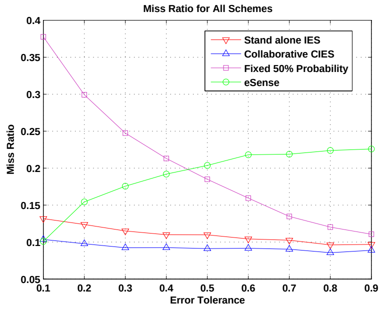

Fig. 17. The Miss Ratio with different error tolerance

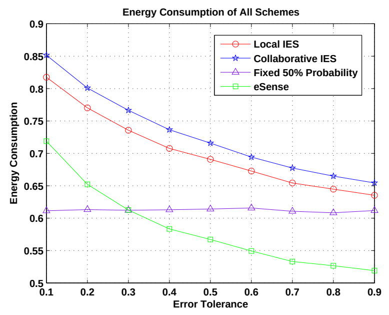

Fig. 18. The energy consumption with different error tolerance

We also compare the performance of our approaches to the state-of-the-art eSense approach [7] ( eSense has only local error control) and an additional benchmark. The benchmark is to set with a random 50% probability rate for $P_m$ for each time instance $m$. We implement the principle of eSense to control the probability for miss ratio performance into our simulation system. The performance of all approaches are demonstrated below,

- **The error rate comparison**. The error performance is demonstrated in Figure 16. As we will see, the benchmark will not be affected too much by the setting of error tolerance. Although the error rate for eSense does not increase too much due to the increase of error tolerance, its error rate is much higher than both $IES$ and $CIES$ approaches. Note that the error rate determined by eSense can not satisfy the error rate boundaries while both stand-alone $IES$ and collaborative $CIES$ approaches meet the requirement.
- **The miss ratio comparison**. The miss ratio, as shown in Figure 17, for eSense continues to increase with the increase of error tolerance, while our methods seem to be stable. The hike in eSense is due to the reduction in sensitivity to the change when the threshold or risk tolerance increases. The higher the risk tolerance, the less sensitive eSense is to the data change, leading to a higher miss ratio. Our system can guarantee the absolute error rate, which will keep tracking the past error and adjust accordingly. Therefore, our approaches will not experience a similar hinderance.
- **The energy consumption comparison**. The energy consumption shown in Figure 18 indicates that eSense con-

| Algorithms | MR($e_t = 10\%$) | EC($e_t = 10\%$) | MR($e_t = 20\%$) | EC($e_t = 20\%$) | MR($e_t = 30\%$) | EC($e_t = 30\%$) |
|---|---|---|---|---|---|---|
| eSense(dynamic) | 0.09 | 0.71 | 0.14 | 0.67 | 0.18 | 0.62 |
| IES | 0.09 | 0.82 | 0.098 | 0.76 | 0.101 s | 0.72 |
| CIES | 0.075 | 0.85 | 0.077 | 0.80 | 0.081 s | 0.76 |

TABLE I
THE PERFORMANCE COMPARISON WITH ESENSE

sumes slightly less than 15% of that of our $CIES$. However, the stability of our system is much better than eSense as the maximum variance of energy consumption is just 12% compared to almost 100% in eSense. These achievements only cost 20% more energy consumption than eSense, which is acceptable in most applications. In some situations, rare event detection is as important as the lifetime management.

To see compare the performance more directly, we list the performance metrics of our design with those reported in $eSense$ [7]. The results are shown in Table I.

Based on comparisons, we conclude that $CIES$ can outperform the eSense with respect to the miss ratio and total error rates. The results also confirm that the error-bounded scheduling limitation, which is missing in eSense, is achieved in our approach.

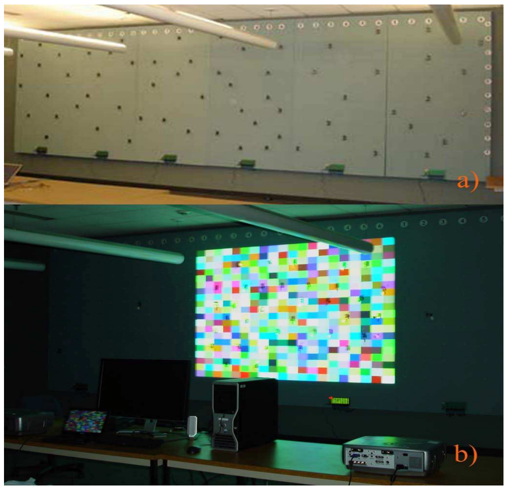

Fig. 19. Testbed and snapshot of experimental data event a) shows the testbed b) shows the operation setup

## E. System Implementation

The architecture has been implemented on our newly constructed test-bed, shown in Figure 19. This test-bed is composed of six 4-foot by 8-foot boards. Each board in the system can be used as an individual sub-system, which is separately powered, controlled and metered separately. Three Hitachi CP-X1250 projectors, connected through a Matorx Triplehead2go graphics expansion box, are used to create an ultra-wide integrated display on those six boards. The design has been implemented on Berkeley TinyOS/Micaz platform, with compiled image occupies 17,076 bytes of code memory and 549 bytes of data memory. Sensor nodes are divided into several groups according to space proximity as planned. Both random and controlled scanning light patterns are created to emulate the light intensity change in environment and then projected onto test-bed with three projectors switched on simultaneously. Those sensing data are broadcasted back to a powerful base-station where complex and high-power consumption computation especially whole network level data reduction is performed. The experimental results of *Non-collaborative IES* and *CIES* collected allow further analysis to optimize the overall system.

The data of our energy consumption to error tolerance sensitivity can be viewed in Figure 20. As presented, different from the fixed probability baselines, the *IES* method can leverage its energy conservation with different error tolerances.

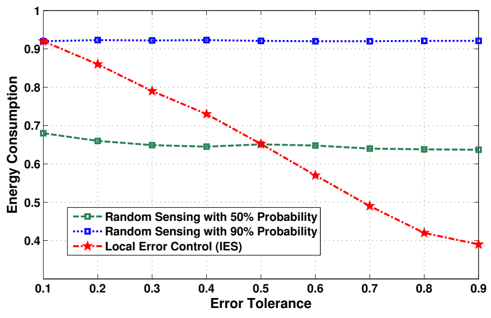

Fig. 20. The measured energy consumption of $IES$ and its comparisons

## VI. RELATED WORK

Scheduling control has been an effective method in wireless sensor platforms to improve energy efficiency. It allows networked nodes to reduce their transmitting and sensing power while preserving sensing quality. A common technique to reduce power is local data control, e.g., reducing the duty cycle (the active duration) of sensors by turning them off while using a prediction model to estimate the actual data at each sensor node [19], [20]. In general, a higher duty cycle leads to a better prediction accuracy, but consumes more energy. In contrast, a reduced duty cycle is preferred due to its lower energy consumption and reduced data traffic within the network, but this may decrease the prediction accuracy.

Traditionally, complex and dedicated models have been used in local data control, aiming to determine if the model is accurate enough to ensure high precision [21], [22]. For example, the Dual Kalman Filter [23], typically used in target tracking applications has been used to predict the actual locations of objects precisely. In [22], empirical analysis results are used to reveal the relationship between the configuration parameters and the quality of the tracking application. It shows that empirical models [14], [24], [25] can be effectively applied without sacrificing the data prediction accuracy. One useful insight from an empirical modeling approach is that data correlation can be utilized for other purposes [26], [27], such as monitoring applications. In eSense [7], a stochastic sensing algorithm that computes the sensor switching probability at each sampling cycle is introduced. It determines the lower and upper bounds of sensing probability to satisfy missing ratio constraints, a metric to determine the percentages that the prediction model output will violate the data performance requirement. In actual situations, however, this kind of approach cannot necessarily characterize the volatile nature of the environment, caused by the insensitivity of prediction model to sharp changes in natural environment [28], thus leading to inefficient data prediction.

Another category of data control is the implementation of a distributed data management scheme. Data aggregation approaches have been widely acclaimed as useful techniques to reduce communication overhead in sensor networks [29], [20]. However, there has been little cooperation between sensor nodes. Although those approaches offer data management mechanisms which reduce the error and energy cost of sensing activities, they fail to improve the system performance through network coordination. A mechanism of node cooperations, together with local data management, can provide the opportunity to accurately associate networking nodes for higher data accuracy and increased network capability, e.g. detection of the rapid environmental change.

## VII. Conclusions

In this paper, we have presented a stochastic sensing algorithm to reduce energy consumption. Our approach uses the data correlation between nodes to reduce the error rate for prediction model performance. Observed correlations between nodes have been used to estimate the neighboring nodes' errors, and to adjust their operation accordingly. The measurement and simulation results show that system prediction error remains within the specified error tolerance while saving up to 60 percent of the required energy. For our future work, we will evaluate the energy performance of individual sensor network components so that the algorithm can be further optimized.

## References

[1] T. He, S. Krishnamurthy, L. Luo, T. Yan, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh, "VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance," *ACM Transaction on Sensor Networks*, To appear.

[2] H. Luo, J. Wang, Y. Sun, H. Ma, and X.-Y. Li, "Adaptive sampling and diversity reception in multi-hop wireless audio sensor networks," in *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, 2010, pp. 378 –387.

[3] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks," in *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.

[4] X. Bai, C. Zhang, D. Xuan, and W. Jia, "Full-coverage and k-connectivity (k=14, 6) three dimensional networks," in *INFOCOM 2009*.

[5] Z. Yun, X. Bai, D. Xuan, T. H. Lai, and W. Jia, "Optimal deployment patterns for full coverage and k-connectivity (k $leq$ 6) wireless sensor networks," in *INFOCOM 2009*.

[6] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage Problems in Wireless Ad-Hoc Sensor Networks," in *Proc. of INFOCOM*, 2001.

[7] A. S. J. Liu, H.; Chandra, "esense: energy efficient stochastic sensing framework for wireless sensor platforms," in *IPSN 2006*.

[8] C. yee Chong, Ieee, S. P. Kumar, and S. Member, "Sensor networks: evolution, opportunities, and challenges," in *Proceedings of the IEEE*, 2003, pp. 1247–1256.

[9] "Atmega128l datasheet," in *http://www.atmel.com/dyn/resources/*, USA, 2008.

[10] J. Polastre and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[11] W. Gu, X. Bai, S. Chellappan, D. Xuan, and W. Jia, "Network decoupling: a methodology for secure communications in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 12, pp. 1784 – 96, 2007/12/.

[12] B. Matt and M. Mundy, "Designing efficient and resilient tactical sensor network neighborhood keying algorithms," Nassau Inn, Princeton, NJ, USA, 2007//, pp. 1 – 7.

[13] A. Jindal and K. Psounis, "Modeling spatially correlated data in sensor networks," *ACM Trans. Sen. Netw.*, vol. 2, no. 4, pp. 466–499, 2006.

[14] C. L. Y.-X. H. N. Xiong, "An energy-efficient dynamic power management in wireless sensor networks," in *Parallel and Distributed Computing, 2006. ISPDC06*.

[15] Q. Zhang, Y. Gu, T. He, and G. Sobelman, "Cscan: A correlation-based scheduling algorithm for wireless sensor networks," *Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on*, pp. 1025–1030, April 2008.

[16] V. Rajamani and C. Julien, "Blurring snapshots: Temporal inference of missing and uncertain data," in *PerCom*, 2010, pp. 40–50.

[17] R. Graham and J. Cortes, "Spatial statistics and distributed estimation by robotic sensor networks," in *American Control Conference (ACC), 2010*, 30 2010.

[18] "Wisconsin-minnesota cooperative extension agricultural weather page," in *http://www.soils.wisc.edu/wimnext/*, University of Minnesota,Twin Cities, MN, United States, 2008.

[19] A. Mainwaring, J. Polastre, R. Szewczyk, D. E. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *Proc. of the ACM Workshop on Sensor Networks and Application (WSNA)*, September 2002.

[20] I. Solis and K. Obraczka, "in-network aggregation trade-offs for data collection in wireless sensor networks," *International Journal of Sensor Networks*, 2006.

[21] R. Mangharam, R. Rajkumar, S. Pollin, F. Catthoor, B. Bougard, L. Van der Perre, and I. Moeman, "Optimal fixed and scalable energy management for wireless networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 1, march 2005, pp. 114 – 125 vol. 1.

[22] Q. Qiu, Q. Wu, D. Burns, and D. Holzhauer, "Lifetime aware resource management for sensor network using distributed genetic algorithm," Tegernsee, Germany, 2006//, pp. 191 – 6.

[23] A. Jain, E. Chang, and Y. Wang, "Adaptive stream resource management using kalman filters," 2004.

[24] C. L. A. M. R. Passos, R.M.; Coelho, "Dynamic power management in wireless sensor networks: An application-driven approach," in *Wireless On-demand Network Systems and Services, 2005. WONS 2005*.

[25] H. C. S. K. Wang, "An adaptive hybrid dynamic power management method," in *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*.

[26] S. Pattem, B. Krishnmachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," in *IPSN*, 2004.

[27] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "On network correlated data gathering," 2004.

[28] S. Goel and T. Imielinski, "Prediction-based Monitoring in Sensor Networks: Taking Lessons from MPEG," *ACM Computer Communication Review*, vol. 31, no. 5, October 2001.

[29] G. Xing, T. Wang, W. Jia, and M. Li, "Rendezvous design algorithms for wireless sensor networks with a mobile base station," in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '08. New York, NY, USA: ACM, 2008, pp. 231–240. [Online]. Available: http://doi.acm.org/10.1145/1374618.1374650