Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 286981, 17 pages
doi:10.1155/2012/286981

Research Article

Adaptive WSN Scheduling for Lifetime Extension in Environmental Monitoring Applications

Jong Chern Lim and Chris Bleakley

UCD Complex and Adaptive Systems Laboratory, UCD School of Computer Science and Informatics, University College Dublin, Ireland

Correspondence should be addressed to Jong Chern Lim, jongchern@gmail.com

Received 15 June 2011; Accepted 27 August 2011

Academic Editor: Yuhang Yang

Wireless sensor networks (WSNs) are often used for environmental monitoring applications in which nodes periodically measure environmental conditions and immediately send the measurements back to the sink for processing. Since WSN nodes are typically battery powered, network lifetime is a major concern. A key research problem is how to determine the data gathering schedule that will maximize network lifetime while meeting the user's application-specific accuracy requirements. In this work, a novel algorithm for determining efficient sampling schedules for data gathering WSNs is proposed. The algorithm differs from previous work in that it dynamically adapts the sampling schedule based on the observed internode data correlation as well as the temporal correlation. The performance of the algorithm has been assessed using real-world datasets. For two-tier networks, the proposed algorithm outperforms a highly cited previously published algorithm by up to 512% in terms of lifetime and by up to 30% in terms of prediction accuracy. For multihop networks, the proposed algorithm improves on the previously published algorithm by up to 553% and 38% in terms of lifetime and accuracy, respectively.

1. Introduction

Wireless sensor networks (WSNs) consist of nodes which detect and track real-world quantities [1]. Nodes are autonomous and are able to self-organize into intelligent networks. Each node consists of a microcontroller, memory, a radio transceiver, and sensors. Most WSN nodes are battery powered. The limited supply of energy means power consumption is a major issue in WSNs. In most applications, the radio transceivers are the largest consumers of energy [2]. Consequently, much research has been conducted on reducing the amount of time that the radio is on [3–5].

An important application area for WSNs is environmental monitoring [1]. Environmental monitoring applications require that a physical quantity is periodically measured and the measurements are relayed across the network to the base station, or sink, for processing. In many cases, the base station must maintain an up-to-date (online) view of the physical quantity being measured. Thus measurements must be transferred to the sink as soon as they are available [6–8]. WSN measurements of data, such as temperature, humidity, air pressure, wind speed, nitrogen dioxide, and light, often exhibit internode data correlation and strong temporal correlations between different sampling times at the same node [9–12]. Knowledge of these correlations can be exploited to reduce the number of measurements needed to meet the application-specific sensing accuracy requirements.

Figure 1 shows temperature readings taken from two nodes in an environmental monitoring deployment in a university campus. The figure shows that the data is correlated between the two nodes. Also from 17:00 onwards, a strong temporal correlation begins to emerge in the data. Figure 2 shows the results when the number of transmitted samples is reduced by 25%, with every skipped sample being temporally predicted from previous readings. The results show that the temporal predictor shows good accuracy from 17:00 onwards. In Figure 3, rather than having node 1 transmit data samples, only the readings from 25% of the nodes within the network are used at any one time to predict the

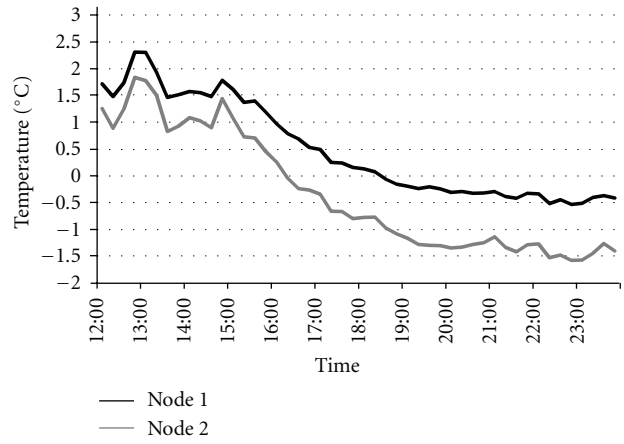


Figure 1: Temporal and internode data correlation of two nodes.

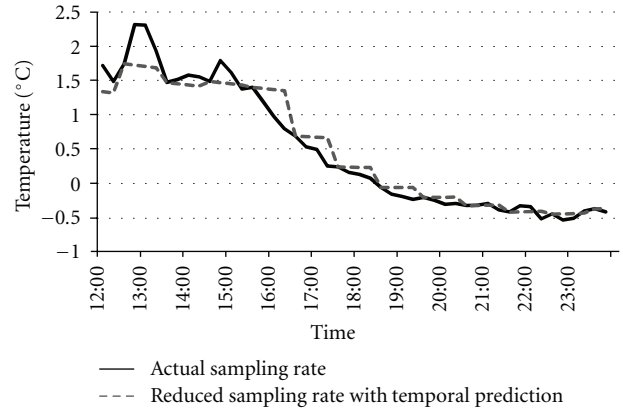


Figure 2: Performance of temporal predictor.

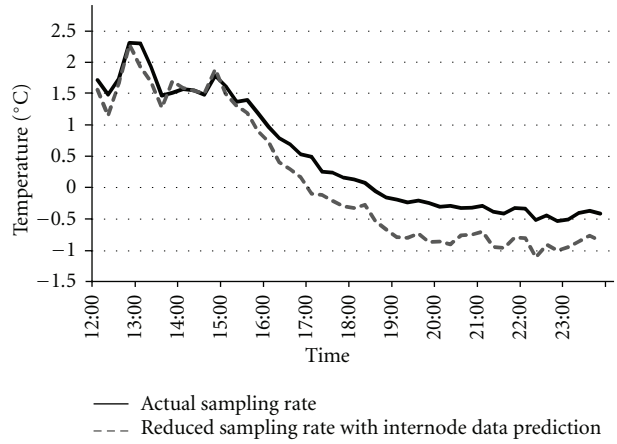


Figure 3: Performance of internode data predictor.

samples (internode data prediction). The results show that the internode predictor works well between 12:00 and 15:00. In this paper an algorithm which takes advantage of both temporal and internode correlation is proposed to reduce the number of transmitted samples at the cost of an application-specific acceptable error.

Clearly, there is a tradeoff between sensing accuracy and lifetime [13, 14]. In general, it can be said that improved accuracy requires collection and transmission of a greater number of sensor measurements which, in turn, means shorter network lifetime. The efficiency of a particular data collection schedule depends on the characteristics of the data being collected. These characteristics vary with time. Hence, the natural question arises: *for a given environmental monitoring application, how can the data gathering schedule be determined and dynamically adapted so as to maximize network lifetime while still meeting the application accuracy requirements*?

In this work, we propose a new adaptive scheduling algorithm for WSNs which can be used in environmental monitoring applications. The algorithm determines the sampling schedule based on user-specified accuracy goals, network connectivity, and a preliminary data collection phase. During preliminary data collection, data is collected from all nodes at the full sampling rate. The preliminary data is divided into training and evaluation data sets. The training data is used to model the temporal and internode data correlation. The evaluation data is used to assess the performance of various candidate scheduling strategies. The models developed in the training phase are used to impute data which is not scheduled for collection according to the candidate strategy. The results of the imputation are compared with the measured data. The schedule which meets the user's accuracy requirements and maximizes network lifetime is deemed to be the most efficient and is applied to the network during the operational phase.

The algorithm supports schedule adaptation to allow for the time-varying nature of the data relationships. Firstly, the algorithm divides the day into a number of time periods or slots. A different subschedule is allowed in each slot. This allows the algorithm to adapt to the differing degrees of correlation present in the data at different times of the day, for example, midnight versus midday. Secondly, the accuracy of imputation is assessed during the operational phase. If the accuracy drops below the user-specific accuracy requirements, the slot is retrained and the subschedule updated. This allows the overall schedule to track long-term changes, such as the lengthening of daytime during spring.

The algorithm differs from previous work in that it supports dynamic adaptation of schedules. The algorithm supports subsampling and round-robin subsetting scheduling strategies. Variants of the algorithm are proposed for two-tier and multihop networks. The performance of the algorithm is assessed by simulation using real-world data sets. The algorithm is shown to significantly extend network lifetime when compared with a previously published scheduling algorithm. In terms of the round-robin subsetting algorithm proposed herein, it is different from coverage-based subsetting algorithms [15–17] in that it uses a data similarity metric rather than physical distance to measure correlation when forming subsets. The benefit of doing this is explained in Section 2.

The remainder of this paper consists of five sections. Section 2 describes related work. This is followed by an

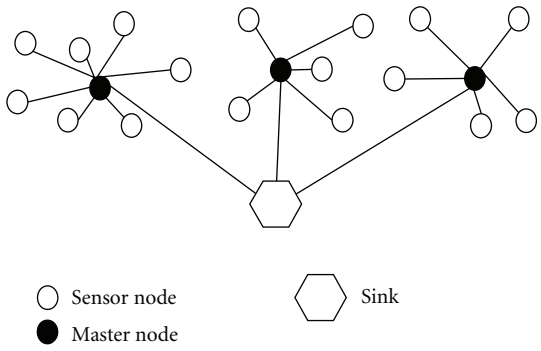


Figure 4: Two-tier network.

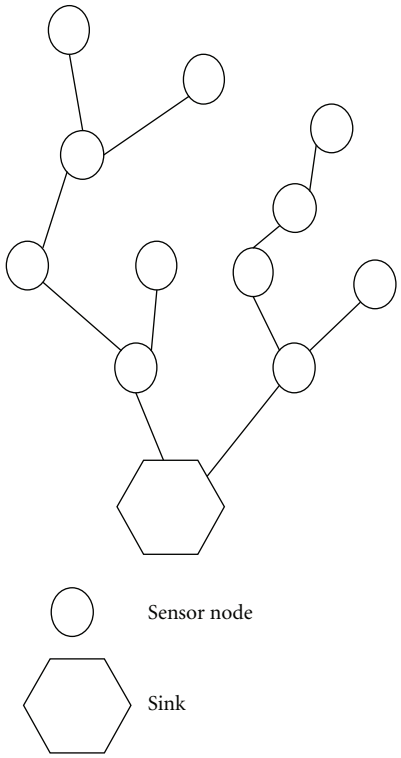


Figure 5: Multihop network.

explanation of the problem in Section 3. In Section 4, the proposed algorithm is described. In Section 5, the experimental method is described. In Section 6, the results and their implications are provided. Finally, the paper ends with conclusions.

2. Related Work

Two network topologies are commonly used for WSN applications: two-tier and multihop networks. Figures 4 and 5 show an example of a two-tier network and a multihop network. In the two-tier case, all battery-powered nodes have direct communication links with mains-powered nodes (master node) which can communicate data to the sink. In the multihop case, only the sink is mains powered and all communication must be routed to it via battery-powered nodes. In the two-tier case, power consumption per node is proportional to the number of measurements per unit time. In the multihop case, power consumption per node is, in the conventional case, not proportional to the number of measurements per unit time, since the routing nodes must be on all of the time. However, in recent research, a number of authors have proposed cross-layer network protocols in which network availability is optimized so that it closely matches the application data transmission requirements [18, 19]. This approach, assumed herein, significantly reduces energy consumption and means that the power consumption per node is proportional to the number of measurements per unit time in the multihop case as well.

The scheduling algorithm proposed herein is targeted at environmental monitoring applications in which all of the data is immediately sent back to the sink. Since all of the data is sent to the sink for data gathering purposes, it makes sense to use this data for centralized scheduling as well. This obviates the need for energy-inefficient intranode schedule negotiation and allows for exploitation of multihop data correlations. In addition, much more computationally complex scheduling algorithms can be used at the sink than can be performed on the nodes, further improving performance.

Reducing the amount of data gathered in a WSN can be done by subsampling or subsetting. Subsampling is the process of making measurements less frequently; for example, a subsampling ratio of 2 would increase node sampling periods from 1 minute to 2 minutes. Round-robin subsetting is the process of using only a proportion of the nodes at any one time in a round-robin fashion; for example, a subsetting ratio of 2 would mean that half the nodes are sampled in even-numbered minutes (1, 3, 5, etc.) and the other half are sampled in odd-numbered minutes (0, 2, 4, etc.). In both examples, the energy consumption of the network is halved. The level of accuracy in imputing missing data varies depending on the degree of temporal or internode data correlation. The algorithm proposed in this work uses both subsampling and round-robin subsetting.

A number of publications have dealt with subsampling [20–22]. In all cases, measurements are suppressed, that is, not transmitted, if they can be accurately predicted based on previous measurements. Data suppression can either be *a priori*, before the measurement is taken, or *posteriori*, after the measurement is taken. As will be seen, depending on the data set, sometimes subsetting outperforms subsampling and sometimes vice versa. Hence the proposed approach supports both subsetting and subsampling.

Several publications have proposed algorithms for subsetting. These algorithms can be classified according to whether the subsetting decision is made based on the geographical coverage of the nodes or based on the data sensed by the nodes. Coverage-based schemes attempt to schedule nodes such that the entire area of interest is covered by the fewest sensor nodes [15–17]. The difficulty with this approach is that when obstacles are present within the area being monitored, sensor readings will not be well correlated with location [23]. In such cases the predominantly assumed

Table 1: Previous algorithms: main features.

Algorithm	Reference	Two-tier	Multihop	Centralized/distributed	Round-robin subsetting	Subsampling	Adaptive scheduling
CAG	[19]	×	√	Distributed	×	√	×
GUPTA	[18]	×	√	Semidistributed	×	×	×
KEN	[24]	×	√	Distributed	×	√	×
SeReNe	[29]	×	√	Centralized	×	×	×
RRC	[25]	√	×	Centralized	√	×	×
SS-MH/SS-2T	Proposed Method	√	√	Centralized	√	√	√

disc-shaped sensing radius no longer holds true. For example, two sensors may be close together but be on different sides of a wall. In addition, node location information may not be readily available. Hence, in this work, we focus on data-similarity-based approaches. Another benefit of using a data similarity/correlation approach is that it can detect correlation changes in the environment over a long period of time. In this paper it is shown that as internode data correlations change, remodeling/retraining has to be done to maintain high accuracy in data imputation.

A number of methods have been proposed for subsetting based on data similarity. These methods can be grouped according to whether they use a centralized or distributed approach. In the centralized approach, the sink determines the sampling schedule whereas in the distributed approach, the nodes themselves decide on the subsets. The disadvantage of the distributed approach is that, if subsets are large, initializing and maintaining them requires a significant amount of internode communication, as in KEN [24]. As a consequence, contour maps and CAG [19] limit the range of subsets to one hop. The disadvantage of this is that long-distance correlations cannot be exploited. Furthermore these subsetting algorithms do not use a round-robin scheme thus achieving poor load balancing.

Herein we compare the proposed approach with the algorithm (which is named GUPTA in this paper) described in [18]. The GUPTA algorithm uses a data-driven approach, and two-tier and multihop versions are described. Unlike the algorithm proposed herein, the GUPTA method does not consider temporal correlations, adaptive scheduling, load balancing, or slotted scheduling. In the multihop version, the GUPTA algorithm is semidistributed because even though nodes make individual decisions whether to join a subset, it requires a centralized data gathering phase in order for all the nodes to gather training data from their neighbors.

In order to achieve load balancing for two tier networks, two systems have been previously proposed which incorporate round-robin subsetting [25, 26]. The system proposed in [25] converges slowly, forming multiple clusters before finding a satisfactory solution. This means that the system produces a significantly higher number of schedules thus making it difficult to maintain. The system described in [26] was developed by the authors of this paper as a prototype. The version described in this paper has a number of improvements. In addition to that we propose a novel network optimized load-balanced subsetting for multihop networks.

Two systems have been previously described which use both subsetting and subsampling-KEN [24] and contour maps [27]. Unlike the proposal described herein, these algorithms do not perform any network level optimization, in the sense that nodes will still have to switch on their radios periodically to listen for packets as well as to relay packets even when they have no readings to send. Furthermore round-robin subsetting is not used.

Combining statistical WSN data models with probabilistic queries to improve the cost-effectiveness of WSN queries was investigated in the BBQ system [28]. However, BBQ focuses on multiple one-shot queries over the current state of the network, rather than continuous data gathering. In [29] SeReNe, a scheduling algorithm for answering queries is proposed. Similar to BBQ and the proposed method herein, it first gathers historical sensor readings. Through clustering SeReNe builds a subset of representative nodes to answer queries. The disadvantage of that is that for long-term queries SeReNe does not employ a round-robin scheme to achieve load balancing. In [30] the originators of SeReNe briefly discuss possible ways to adapt the model over a long period of time, but this was not evaluated. KEN uses data models as well to answer queries. KEN and SeReNe are similar in the sense that they are push-based methods whereas BBQ is a pull-based method. Herein, the user sets a probabilistic accuracy target *a priori* and possible schedules are assessed with respect to the target prior to their application.

A comparison between the various data-similarity-based scheduling algorithms that have been proposed is provided in Table 1. The algorithm proposed herein is the first to support schedule adaptation and round-robin subsetting.

3. Problem Statement

The goal of the scheduling algorithm is to determine the network sampling schedule which minimizes network communication for the worst-case node while ensuring that application-level accuracy requirements are met. The reason for minimizing communication by the worst-case node is to maintain load balancing thus enabling the network to continuously gather data from all nodes within the network continuously for a longer period of time. Even though sensor

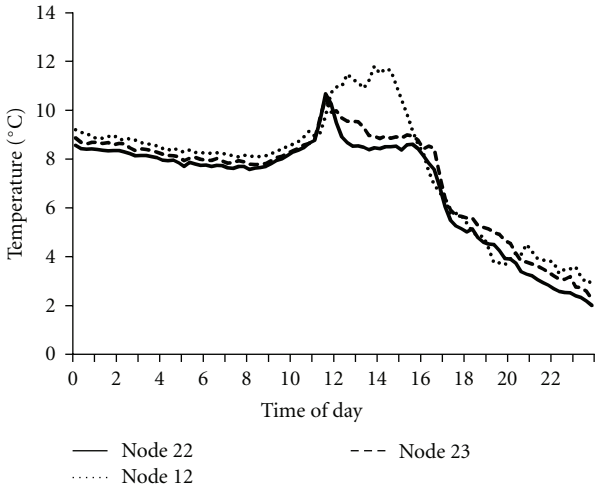


Figure 6: Data relationship of three nodes (temperature-LUCE deployment).

data of dead nodes can still be imputed, because the node is dead, validation and retraining of the predictors cannot be done when needed.

The user defines the accuracy requirement by setting a limit on the average probability (P_{lim}) of errors greater than a specified threshold (E_{lim}). For example, the user might require that 95% of reported measurements have an error of less than 0.5°C. In the case of measured values, the error e is equal to zero. In the case of imputed values, the error may be greater than zero. The goal of the algorithm is then to determine the schedule S_{ch} which minimizes the number of packets N_p transmitted by the worst-case node such that the probability $p(e)$ of errors less than E_{lim} is greater than P_{lim}

$$S_{\text{ch}} : \min\left(N_p\right) \quad \text{s.t. } p(e < E_{\text{lim}}) > P_{\min}. \tag{1}$$

As stated previously, data correlations can be exploited in order to impute the missing values. In most previous work, these correlations are assumed to be static. Figure 6 shows the variation of temperature at three nodes over a day in a real-world dataset. Clearly the rate of change and internode data correlations are dependent on the time of day. Thus a scheduling algorithm should account for the fact that data correlations drift during the day and, for best performance, should use different subschedules at different times of the day. In addition, over long periods of time, temporal and internode data correlations can vary. Thus, imputation becomes less accurate. This deterioration in performance should be detected and the models retrained.

When subsetting, it is desirable that the subsets are disjoint and operate in a round-robin fashion so that the network is load balanced. Disjoint subsets are subsets such that for any two subsets C_i and C_j, $C_i \cap C_j = \phi$; that is, every node belongs to only one subset. In the two-tier case, determining disjoint subsets which provide accurate imputation of environmental conditions at all nodes is nontrivial. In the multihop case, the problem is more complex since every disjoint set must provide a representative

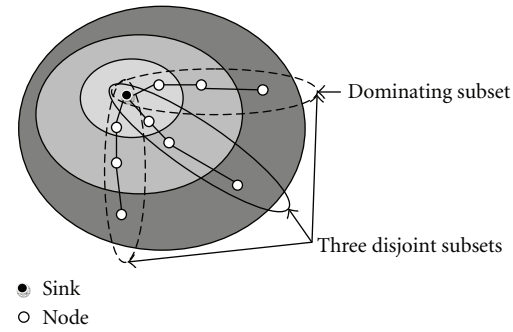


Figure 7: Disjoint subsetting example.

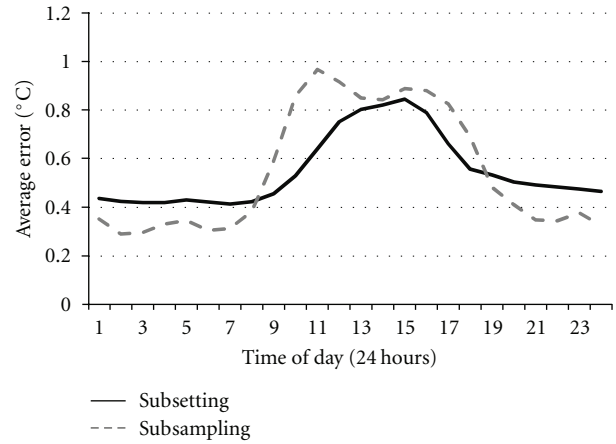


Figure 8: Performance of subsetting and subsampling averaged over 105 days.

node to represent each correlated region while also ensuring connectivity between all the nodes in the subset and the sink. For example, the three disjoint subsets in Figure 7 allow both load-balanced subsetting and continuous connectivity while having each correlated region represented by a node.

Figure 8 shows the performance of subsetting and subsampling with 75% of the data being predicted. Both methods are explained in detail in the following section. The figure shows that both algorithms perform well in the morning and at night. During the afternoon, both algorithms experience a significant loss in performance. Thus, on average, even if the accuracy of the method meets the user's requirements initially, it does not mean that the requirements are met throughout the day. To ensure user requirements are met, the amount of data being predicted during the afternoon has to be decreased. This can be done by reducing the subsampling/subsetting ratio.

4. Proposed Algorithm

In this section we explain the proposed slotted scheduling algorithm with variants for two-tier (SS-2T) and multihop (SS-MH) networks. The following sub-sections provide an overview of the algorithm; explain how schedules are

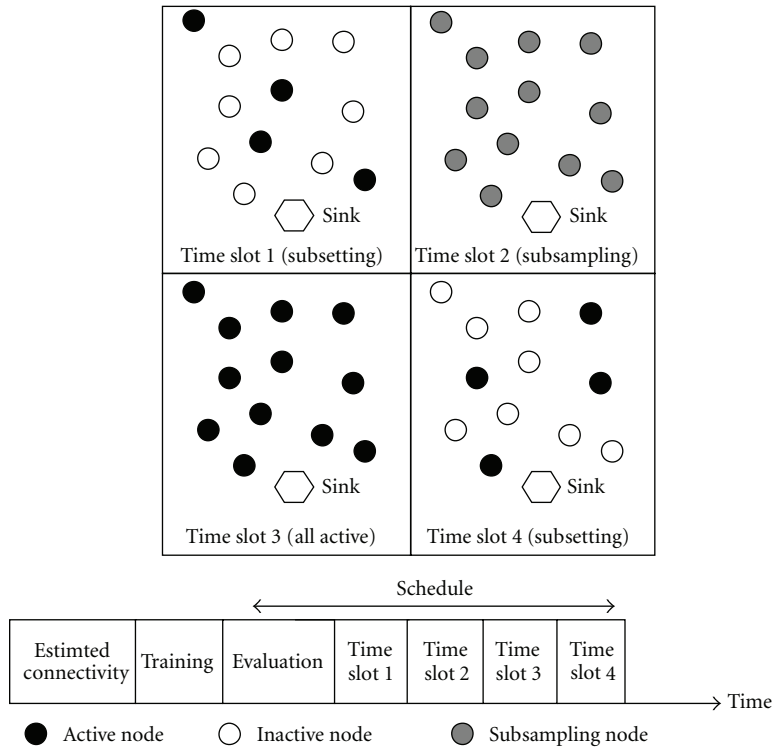


Figure 9: Slotted-scheduler timeline and network activity.

defined; describe how data imputation is performed; explain node-to-subset allocation for round-robin subsetting in both two-tier and multihop networks; explain the schedule selection process and detail the schedule update method.

4.1. Overview. Initially, the slotted scheduler gathers training and evaluation data and, in the multihop case, connectivity information from the network. During training and evaluation data collection, all nodes collect data at the user-specified maximum collection rate and transmit this data back to the sink. At the sink, the training data is used to build models for data imputation. The data from the evaluation phase is then used to assess the performance of various candidate scheduling strategies, that is, various ratios of subsetting and subsampling. The subschedule which meets the user's accuracy requirements and minimizes energy consumption is selected for application to the network in that slot during the operational phase. The selected data collection schedule is transmitted from the sink to the nodes. The network then enters the operational mode and data is collected according to the schedule. Data collected is monitored in order to detect changes in temporal/internode correlation. If changes are detected, the network reenters the training and evaluation phases in order to update the models and schedule.

Figure 9 illustrates how the slotted scheduling algorithm operates. The figure shows a 4-slot schedule with subsetting, subsampling, full rate collection and subsetting in the first, second, third, and fourth slots, respectively. The figure also shows the temporal sequencing of the establishment, training, evaluation, and operational phases. The operational phase is divided into a series of slots which repeats.

Figure 10 shows how subsetting or subsampling is chosen for each time slot. The temperature data plot, taken from the evaluation data, shows that between the times of 7:00 and 17:00 there is limited temporal correlation. The data does show internode data correlation during this period; thus, subsetting is selected for that time period. During this time, nodes 1 and 2 exhibit strong correlation and nodes 3 and 4 show strong correlation. To ensure each inactive node is represented within the active subset by a strongly correlated node, the subsetting algorithm groups nodes 1 and 3 as one subset and nodes 2 and 4 as the other subset. These two subsets are used in a round-robin fashion during the operational phase, as shown in the figure. From 17:00 onwards, the data begins to show strong temporal correlations. During this period, the temporal predictor performs more accurately. Thus subsampling is chosen for use by the scheduler. The figure shows that after 17:00 all nodes are activated for one time slot, and for the subsequent two time slots, nodes are inactive and data is temporally predicted.

4.2. Schedule Description. The schedule is based on the user-specified default data collection period. This is the maximum rate at which data can be collected, that is, with no subsampling or no-subsetting applied. The schedule is divided into a number of slots, or time periods, which span the day. A different subschedule can be specified for each slot. This allows the scheduler to adjust the data collection rate depending on time of day. For example, in a schedule with eight slots, each slot would last for four hours: slot 0 from midnight to 4 AM, slot 1 from 4 AM to 8 AM and so on.

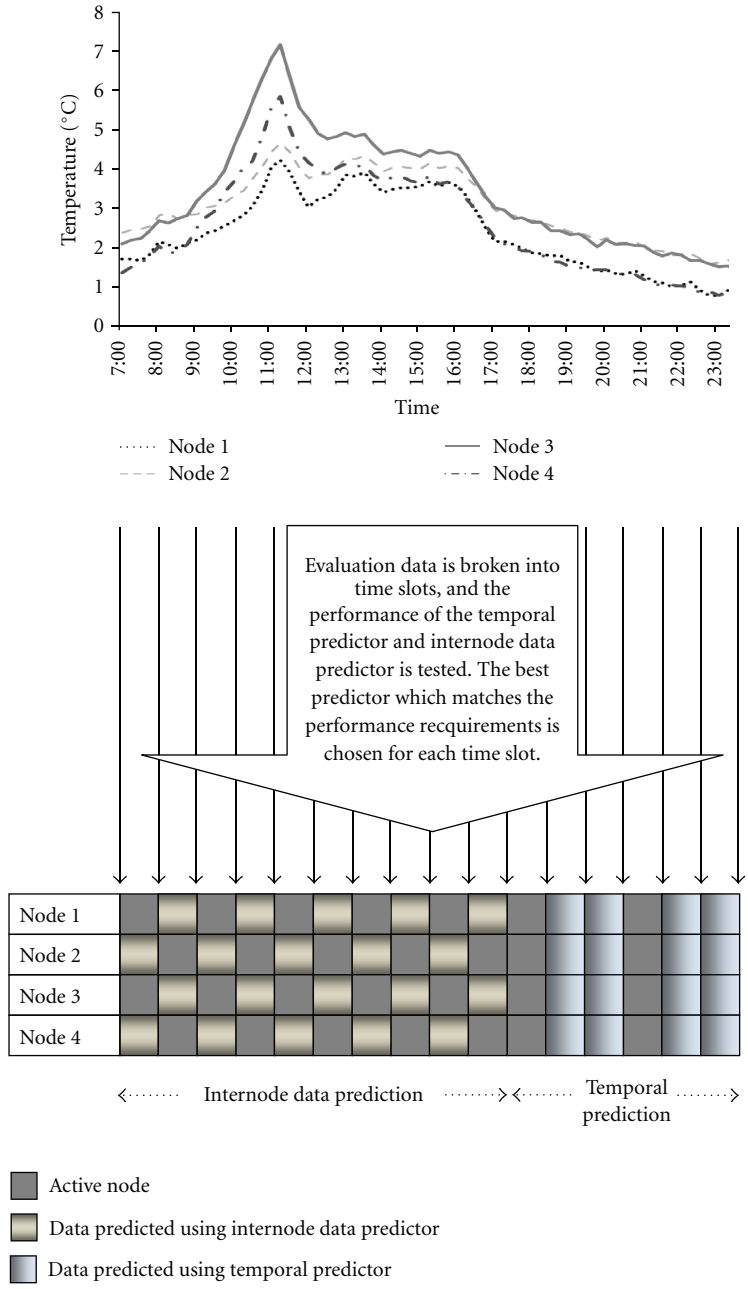


Figure 10: Scheduler overview.

Within each slot, the node sampling subschedule is specified as the sampling rate at which the node is sampled relative to the default collection rate. For example, a subsampling rate of 100% means that a node collects data at the default collection rate. The node sampling offset is used to indicate which data collection round the node starts to be sampled relative to the start of the operational phase.

A node schedule consists of

(i) node id,

(ii) eight bits indicating default data collection period (in minutes),

(iii) eight bits indicating slot length,

(iv) for each slot, three bits indicating the sampling rate, five bits indicating the sampling offset.

The duration of a slot equals the slot length multiplied by the default data collection period. There are eight different sampling rates which can be used with the maximum being 100% and the minimum 12.5%.

For a twenty-four-slot schedule, a single-node schedule (excluding node id) is 26 bytes. In TinyOS (which has a default data packet payload of 28 bytes [31]), the cost of sending the schedules from the sink to the nodes is roughly equivalent in terms of energy consumption to sending one data measurement from all of the nodes to

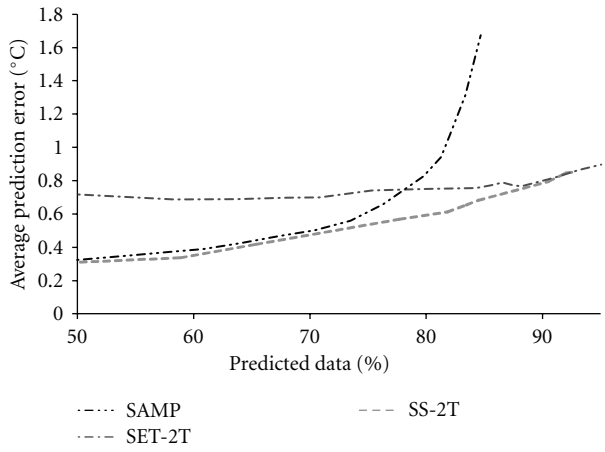


Figure 11: Variation in mean imputation error with percentage of noncollected data.

the sink. Piggybacking and compression schemes can be used to reduce this overhead. Data collection timing can be maintained using node wake-up synchronization [32].

Herein, we refer to data which is scheduled for collection as collected data and data which is not scheduled for collection as noncollected data. Noncollected data must be imputed based on collected data.

4.3. Data Imputation. In the case of subsampling, imputation is performed using linear prediction (LP). The linear predictor determines the coefficients of a forward linear predictor by minimizing the prediction error in the least squares sense based on the training data. During the operational phase, LP is used to estimate the noncollected data as a weighted sum of previous measurements obtained at the same node

$$x_i(i,t) = a(i,1)x_o(i,t-r) - a(2)x_o(i,t-2r) - \cdots - a(p)x_o(i,t-pr), \quad (2)$$

where $x_i(i,t)$ is the current imputed sample at node i at time t, $x_o(i,t-r)$ is the observed (measured) data at node I at time $t-r$, $a(i)$ are the coefficients of the linear predictor, r is the subsampling ratio, and p is the length of the predictor.

In the case of subsetting, only one subset of the network is collected in each data collection round. Given that subset C_i is the operating subset consisting of the nodes $s_1, s_2, \ldots, s_L$, then the predicted value of a node is

$$X_p = \sum_{l=1}^{L} \alpha_l s_l. \quad (3)$$

Given that the training data for a single node and the remaining nodes is o and O, respectively, then the weighted coefficients are

$$[\alpha_1, \alpha_2, \ldots, \alpha_L]^T = \left(O^T O\right)^{-1} O^T o. \quad (4)$$

4.4. Round-Robin Subsetting. To achieve load balancing, every node in the network is allocated to a subset and the number of nodes per subset is constant. The key to accuracy is in allocating the nodes such that every subset contains a set of nodes which accurately represent environmental conditions over the whole network. Novel algorithms have been developed to solve the node allocation problem for subsetting in two-tier and multihop networks.

4.4.1. Two-Tier Networks. In the two-tier case, node-to-subset allocation is achieved by node clustering, followed by subset allocation, and allocation optimization.

Initially, nodes are clustered based on data similarity. Nodes are clustered using a normalized cut (N-cut) clustering algorithm [33] based on an entropy S metric. In this way, nodes with strong data relationships are put in the same cluster

$$S(i,j) = \ln\left(\sqrt{(2\pi e)^2 |\Sigma|}\right), \quad (5)$$

where Σ is the covariance matrix of data obtained from nodes i and j.

After clustering, node allocation is performed. The first node subset is formed by selecting one representative node from each cluster. In this way, the subset consists of nodes which represent the measurements in each cluster. The representative node is chosen as the node with the minimum total entropy $S_{\min}$ within the cluster

$$S_{\min} = \min(S(i,j)), \quad \forall i \in \{1,\ldots,N_c-1\}, \quad \forall j \in \{1,\ldots,N_c-1\}, \quad i \neq j, \quad (6)$$

where N_c are the nodes within the cluster, i is the current, node id and j is the id of the other node.

The second subset is found by excluding the already allocated nodes from the set of available nodes and repeating the representative node selection step. This process is repeated until all of the nodes in the network are allocated to a subset.

The sequential subset allocation process can lead to poor results as the subsets allocated later in the process tend not to perform as well as those allocated earlier in the process. To address this, a genetic algorithm (GA) is applied to optimize the node allocation. First, two subsets are picked at random. Second, one node is chosen from each subset and they are swapped. Third, if the swap causes the sum of the entropy of the two subsets to increase, then the swap is made permanent; otherwise, the subsets revert back to their original states. The full subsetting algorithm is described in Algorithm 1.

Subset allocations and models are generated in this way for a range of subsetting ratios. The allocations are saved for later evaluation, see Section 4.5.

4.4.2. Multihop Networks. In the multihop case, allocation of nodes to subsets is performed in a different way. This is because, in multihop networks, all subsets must provide connectivity between all nodes in the subset and the sink. The algorithm works by growing the maximum number of

```
X = All sensor nodes
i = 1
while X! = ∅ do
    Cluster nodes
    Pick representative node from each cluster
    C_i = Chosen representative nodes
    X = X − C_i
    i++
end
n = number of runs for Genetic Algorithm
while count! = n do
    Pick two random subsets C_p and C_q
    S_total = S_avg_Cp + S_avg_Cq
    C_p_old = C_p
    C_q_old = C_q
    Swap a random node from C_p and C_q
    S_total_new = S_avg_Cp + S_avg_Cq
    if S_total > S_total_new
        C_p = C_p_old
        C_q = C_q_old
    end
    count++
end
```

Algorithm 1: Pseudocode for two-tier round-robin subset allocation.

```
X = All sensor nodes
T_L = Transmission Range Limit
C subsets are formed one for each node x_n within the
T_L of the sink
N_c = number of subsets (equivalent to number of 1 hop
nodes from the sink)
i = 1
X = X − C
while X! = ∅ do
    if i > N_c then
        i = 1
    end
    Pick node x_n which is 1 hop from C_i and has
    highest average Entropy with C_i
    C_i = C_i + x
    X = X − x_n
    i++
end
Save C
while N_c > 2 do
    Combine each subset based on Entropy
    N_c = new number of subsets
    Save C
end
```

Algorithm 2: Pseudocode for multihop round-robin subset allocation.

subsets from the sink based on connectivity information and data similarity.

Using distance criteria, the algorithm determines which nodes are one hop away from the sink. Nodes which are one hop from the sink each form the root of a new subset. Thus the number of new subsets found is directly proportional to the distance criteria. Larger distance criteria will yield a larger number of subsets. The subsets are grown by selecting the nodes according to the following criteria:

(i) being 1 hop away from a node currently in the subset,

(ii) having the highest difference in average entropy between the nodes within the subset.

The subsets are grown in a round-robin fashion. If a subset cannot be grown, then the method continues growing the other subsets. Once the maximum number of subsets has been formed, the method then combines subsets in order to form larger subsets which are better spread over the network. The average difference in entropy between all subset pairs is found. Subsets with the greatest difference are combined. This step is repeated until all subsets have been combined. At each step, the subset allocation is saved for later evaluation, as described in the next sub-section. The multihop subsetting algorithm is fully described in Algorithm 2.

4.5. Selecting the Best Schedule. The performance of all possible subsampling and subsetting strategies is assessed for each slot. The subsampling or subsetting subschedule giving the best performance is selected for application to the network in that slot during the operational phase. The various sub-scheduling options are assessed using the evaluation data. In each case, the noncollected data is imputed and the result compared to the measured data to give the imputation error e

$$e(i,t) = abs(x_i(i,t) - x_o(i,t)). \tag{7}$$

The standard deviation of the error σ calculated over the whole network during the evaluation period is calculated. This is compared to the error target specified by the user. The target standard deviation of the error is calculated by projecting the target error limits (percentage of errors greater than threshold) onto a Gaussian probability distribution and finding the equivalent standard deviation σ_{lim}. Subschedules that lead to error standard deviations in excess of the target $\sigma > \sigma_{\text{lim}}$ are rejected. Since the schedules are load-balanced by construction, the energy consumption of routing is equal in all cases. Thus, the energy consumption is proportional to the number of collected measurements. Therefore, the remaining subschedule with the least number of measurements is selected for application to the network. The final schedule is determined by concatenation of the selected subschedules. If appropriate, the schedule can be compacted by merging consecutive subschedules that are the same, provided that the slot lengths remain equal.

4.6. Schedule Update. During the operational phase, the algorithm monitors the accuracy of the temporal and internode data imputation models. This allows the system to determine if the data characteristics have drifted since the models were last trained. This is done by comparing the

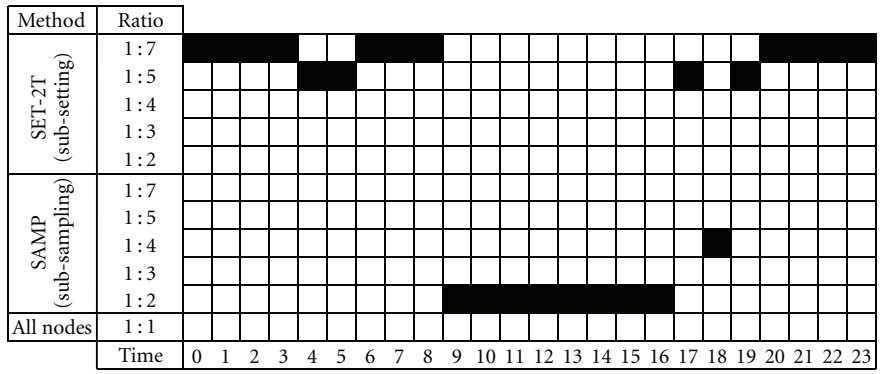
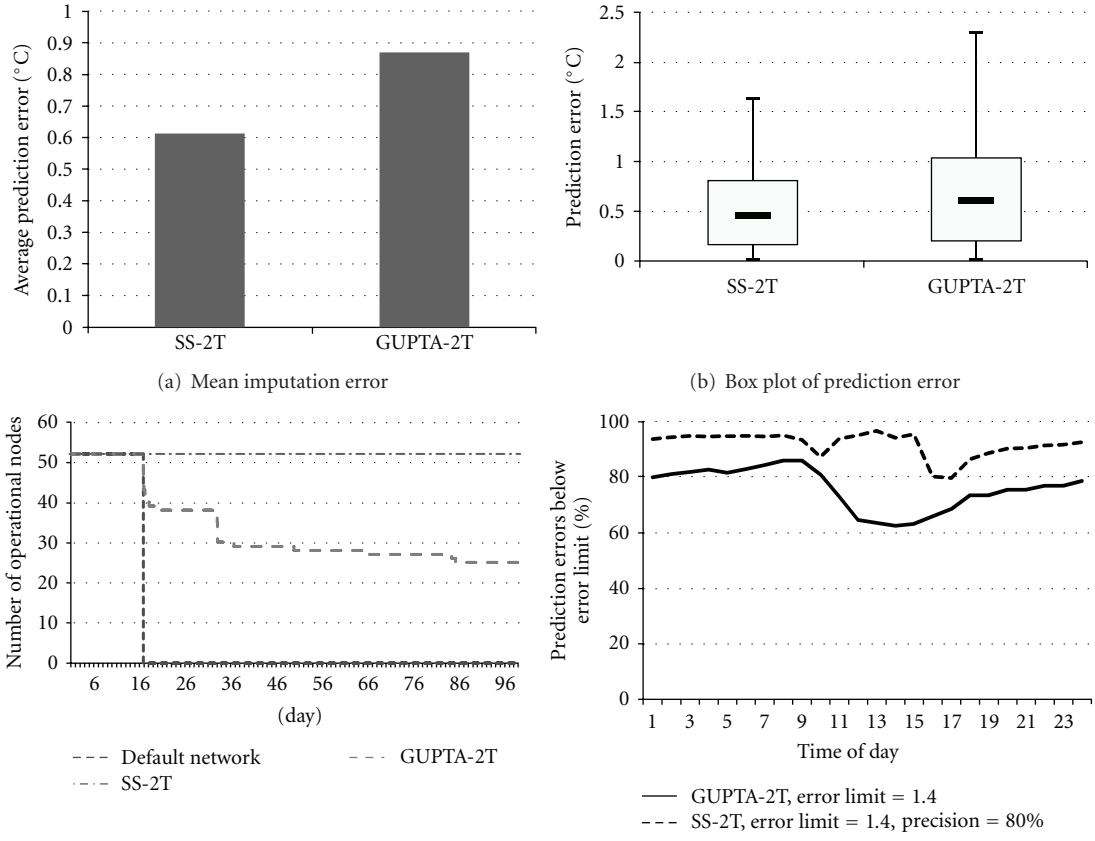


Figure 12: Schedule for ST LUSE dataset, target error of 1.4°C in 80% of cases.



(a) Mean imputation error

(b) Box plot of prediction error

(c) Variation in the number of operational nodes with time

(d) Variation in the number of errors in excess of the limit with time

Figure 13: Performance of scheduling algorithms, ST LUCE dataset, two-tier network.

prediction accuracy seen when the training and evaluation data is used compared to the prediction accuracy seen with the current received sample.

In the case of the temporal model, the model is tested by predicting the current received sample and testing it with the last received sample (which is y samples away). This prediction is done using (2). The error is found between the current received sample and the predicted sample. Next using only evaluation data, the data from the same time slot is predicted with data which is y samples away. A comparison is done between the error found using the current received sample and the error found using the evaluation data. A node is marked when the error difference is above a threshold

limit. When the percentage of marked nodes is above T_{lim} for a duration of D_{lim} days then retraining is triggered.

For the internode model, it is first tested using the current samples received from the nodes of the current operational subset C_i. Using (3) each received sample is imputed using the other received samples at that particular time slot. The error between the predicted value and the actual value for each sample is found. The error results are then compared with the results when the same test is repeated on the evaluation data using the same time slot and the same subset of nodes C_i. A node is marked when the difference between the prediction error using current received samples and the prediction error using evaluation data is above a certain threshold. Similar to the temporal model test, when the limits of T_{lim} and D_{lim} are broken, retraining is commenced.

5. Experimental Method

The algorithm described in the previous section was implemented in Matlab and tested on two datasets taken from the Lausanne Urban Canopy Experiment (LUCE) [34]. Table 2 provides a summary of the datasets.

Results were evaluated in terms of mean imputation error (see (7)), percentage of noncollected data, variation of the number of operational nodes with time, and network lifetime. Mean imputation error is the mean error of the imputed noncollected data. The percentage of noncollected data (PND) is related to the amount of data transmitted and thus to the lifetime of the network. The percentage of data collected and transmitted to the sink is $100\% - PND$. We compare the results for different systems in terms of two definitions of lifetime. The first definition of lifetime is $L_{100\%}$ which is the length of time for which all nodes are alive. The reason for choosing this metric is because when the first node dies, this node can no longer be used for retraining. Thus if the node data correlation with other nodes changes, this cannot be corrected thus rendering the imputed readings from the other nodes void. The second definition of lifetime is $L_{50\%}$ which is the length of time for which 50% or more of the nodes remain alive.

Scheduling algorithms such as [32, 35] reduce idle listening significantly through the proper use of schedules. Such algorithms make the power consumption of sensor nodes closely proportional to the number of transmitted packets. For each simulation done, each sensor node is initialized with a limited battery power. Every transmitted packet is set to consume 1 unit of battery power. We assume the network allows piggybacking thus ensuring that even in the multihop case only a single packet is transmitted by each node during each sampling cycle. Similar assumptions were made in [18].

The performance of the proposed algorithm is compared to that of the default network and to the GUPTA algorithm. In the default network, every node collects data every collection round; that is, all data is collected.

There are two variants of the GUPTA algorithm used herein. GUPTA-2T for two-tier networks and GUPTA-MH for multihop networks. For the GUPTA algorithm, initially when the correlation structure is unknown, all the network

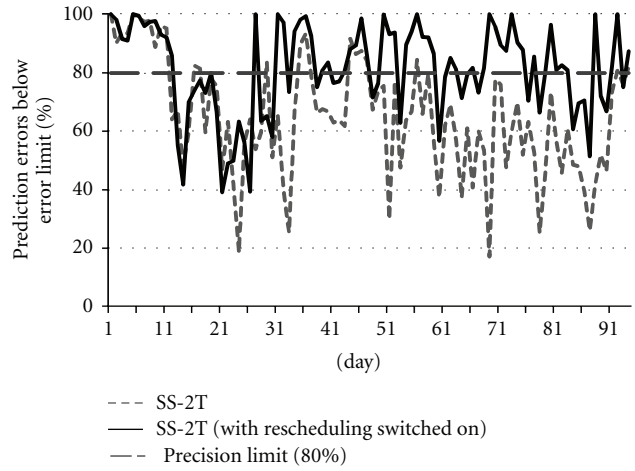


Figure 14: Performance of SS-2T with rescheduling on and off, ST LUCE dataset, two-tier network.

Input: A sensor network with a correlation graph
Output: A correlated dominating set M
BEGIN M = node s_i with largest IM
while $B(\text{new } M, M) > 0$ **do**
  Pick S_i for which $B(M \cup S_i, M)$ is maximum
end

Algorithm 3: GUPTA centralized algorithm.

nodes are periodically involved in transmitting data to the data-gathering node using a communication tree. Using this setup, each node then collects data from its d-hop neighbors using a piggyback scheme. In GUPTA-MH simulations, each node collects 3-hop neighborhood information.

GUPTA-2T algorithm proposed in [18] is used on a multihop network. In [18] during each iteration the number of nodes which can join the connected correlation-dominating set (CCDS) is bounded by the number of hops. As the GUPTA-2T algorithm used herein is used on a two-tier network, the algorithm is no longer bounded by hop count. The benefit of this is that there is a wider selection of nodes which can be added to the operating dominating set.

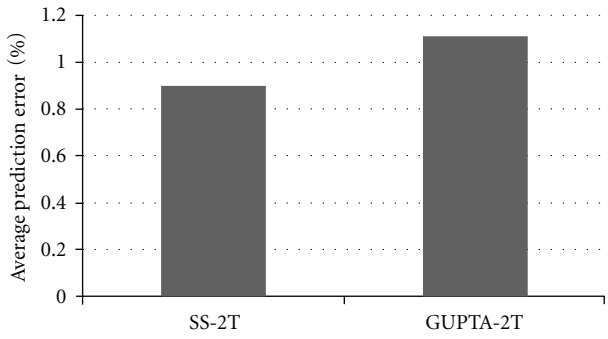
The GUPTA-2T algorithm works by adding nodes which will give the most benefit to the dominating set. This is continuously done till there is no more benefit in adding nodes. Given that IM is the group of nodes which can be inferred by M and new IM the nodes which can be inferred by $M \cup s_i$ (s_i is any node not belonging to M), then the benefit function is $B(M \cup s_i, M) = \text{new IM} - \text{IM}$. The purpose of the benefit function is to maximize the number of inferred nodes thus maximizing the number of sleeping nodes. Pseudocode for GUPTA-2T is shown in Algorithm 3.

For GUPTA-MH, a node s with priority $p(s)$ is marked deleted if the following conditions are satisfied:

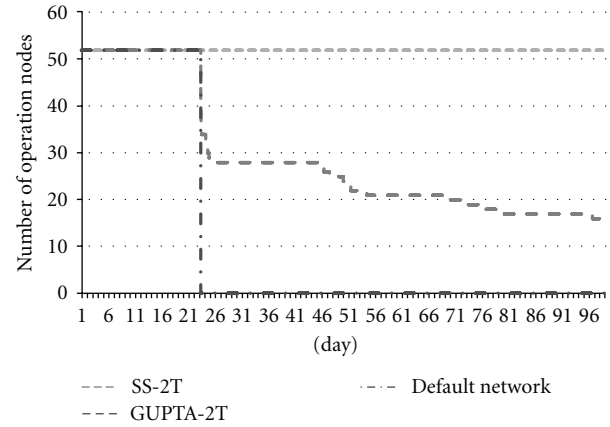
(i) the node s has not been mark selected;

Table 2: Datasets.

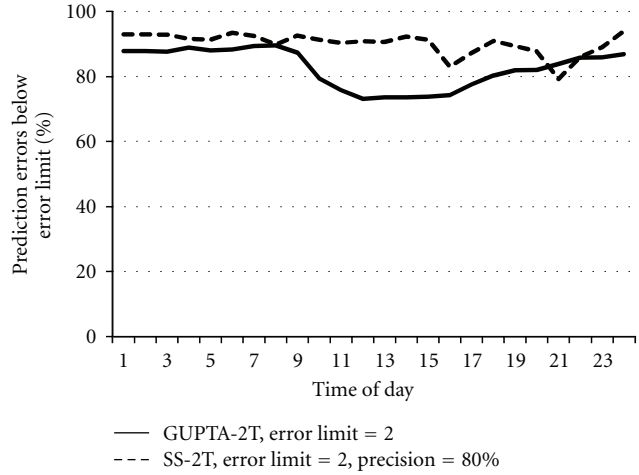
Name	Date	Sampling period	Duration (Days)	Number of nodes	Percentage of missing data	Area
RH LUCE (relative humidity)	1/12/2006	15 minutes	112	52	9.79%	106600 m^2
ST LUCE (surface temperature)	1/12/2006	15 minutes	112	52	7.86%	106600 m^2



(a) Mean imputation error



(b) Variation in the number of operational nodes with time



(c) Variation in the number of errors in excess of the limit with time

Figure 15: Performance of scheduling algorithms, RH LUCE dataset and two-tier network.

(ii) the connectivity of the communication subgraph is not affected by the deletion of the node s;

(iii) there is a correlation edge in the correlation graph such that every node in the set S is either marked selected or has a priority more than $p(s)$.

The score $p(s)$ is the sum of the number of nodes which are correlated with the node s. The more the nodes which can be predicted by s, the higher the $p(s)$.

The number of messages sent during training is not considered in the results as both algorithms require a training phase. For the GUPTA algorithm, the first 14 days are used to build the model. In the case of the proposed algorithm, the first 7 days was used to build the internode/temporal model (training) while the subsequent 7 days was used to assess the performance of various subsetting and subsampling ratios (evaluation). It is assumed that the underlying network is able to handle packet loss. In the multihop case, two nodes are assumed to have connectivity if they are less than 135 meters apart.

In the case of adaptive scheduling, two days of data was used for rescheduling the nodes. The first day is used for a training phase while the second for the evaluation phase. Rescheduling was triggered if 60% T_{lim} of nodes are less than the user-specified error threshold for two days (D_{lim}). When testing, rescheduling nodes with more than 6% of

Table 3: Improvement in lifetime by slotted-scheduler (two-tier) and GUPTA (centralized) with respect to default network (RH LUCE).

Data Set	Algorithm	Error limit	Packet limit	Average prediction error	$L_{100\%}$	$L_{50\%}$
ST LUCE	GUPTA-2T	0.25	2150	0.61	0%	226%
ST LUCE	SS-2)	1.2	2150	0.42	226%	226%
RH-LUCE	GUPTA-2T	0.75	2150	1.1	0%	120%
RH-LUCE	SS-2T	2	2150	0.9	226%	226%

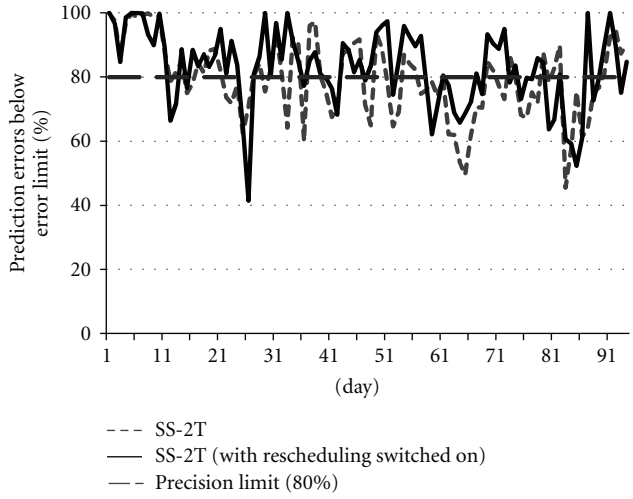


Figure 16: Performance of SS-2T with rescheduling on and off, RH LUCE dataset and two-tier network.

missing data were deleted from the dataset as this impeded rescheduling.

6. Results

This section is divided into two subsections covering the two-tier and multihop cases, respectively. In each section the proposed algorithm is compared with GUPTA, a previously published algorithm. Secondly, the advantages of using retraining with the proposed algorithm are shown.

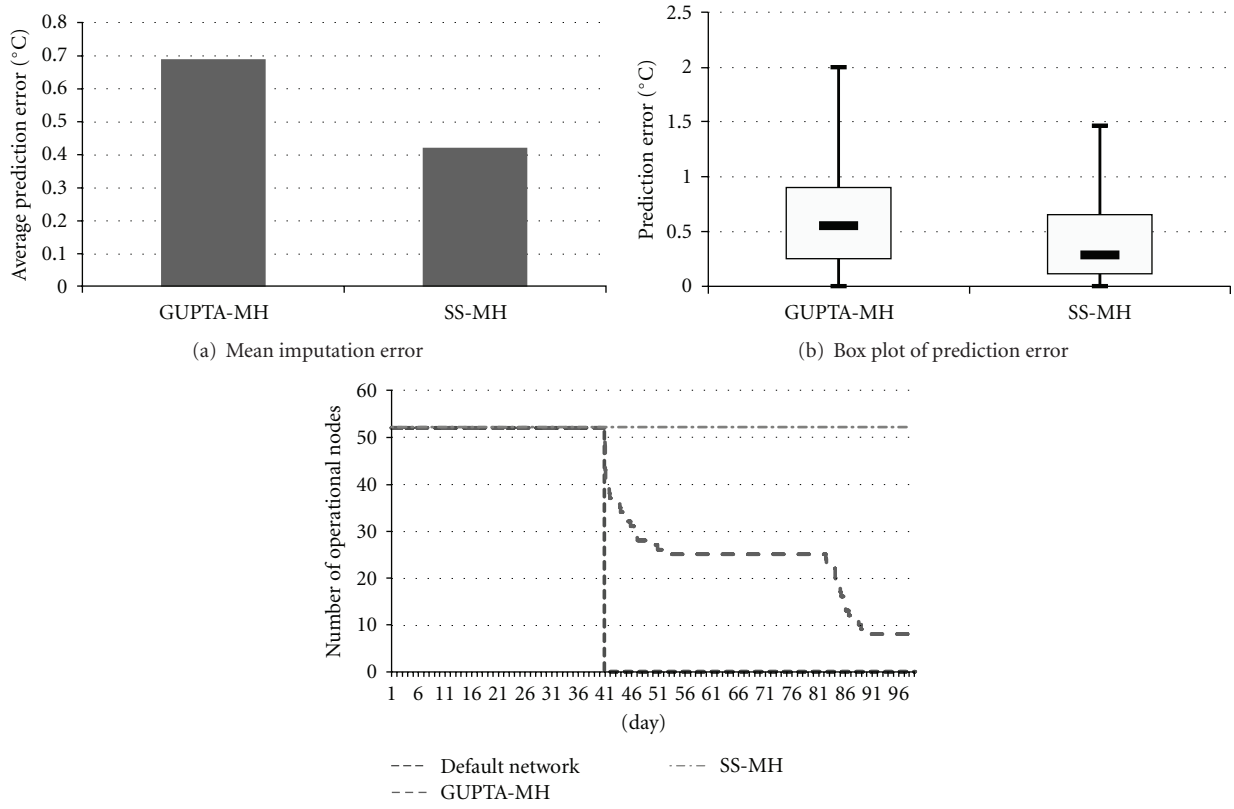
6.1. Two-Tier Network. Firstly the performance of the various subsetting, subsampling, and imputation methods was assessed using the ST LUCE data set and a two-tier network. Figure 11 shows the variation in mean imputation error with the percentage of imputed data for various methods. Three methods were compared—subsampling (SAMP), two-tier subsetting (SET-2T), and the full slotted scheduler algorithm including both subsetting and subsampling (SS-2T). Rescheduling was switched off. For low imputation percentages, subsampling performs better than subsetting. For high imputation percentages, subsetting performs better than subsampling. The proposed slotted scheduling algorithm combines the advantages of subsetting and subsampling and performs best in all cases.

Figure 12 shows the schedule created by the slotted scheduler for an error limit of 1.4°C in 80% of cases. The figure shows that the choice between subsetting versus subsampling as well as the ratio varies during the day depending on the data statistics. Between the times of 00:00 and 09:00 subsetting is scheduled for use. During that period, only one seventh of the nodes were scheduled to sample and transmit at each sampling period for the majority of the duration. From 09:00 till 17:00 (during the day), subsampling is used with a sampling ratio of 1:2. From 20:00 onwards, the scheduler reverts back to the use of subsetting.

The GUPTA and slotted scheduling algorithms were compared using an error limit of 0.25°C and of 1.2°C in 80% of cases, respectively. Rescheduling was switched off. Figure 13(a) shows the mean imputation error of both methods. In terms of prediction accuracy the slotted scheduler performs 29.5% better. A box plot of the prediction error is presented in Figure 13(b). The box plot clearly shows that in terms of the distribution of errors SS-2T performs better as well. Figure 13(c) shows the number of operational nodes over the duration of the simulation for both methods and for the default network. The packet limit was set to 2,150 packets. Using the GUPTA algorithm, nodes start to die much sooner than when using the proposed algorithm. Table 3 shows that in terms of prediction accuracy and lifetime SS-2T outperforms GUPTA-2T. Figure 13(d) shows how the percentage of errors that are in excess of the error limit varies across the time slots during the first week of operation. It can be seen that, for the GUPTA algorithm, the number of errors varies significantly over the slots. The proposed algorithm performs within the 80% precision limit (P_{lim}) for all time slots.

Figure 14 compares the performance of SS-2T with rescheduling on and off. The initial loss in performance in both cases (days 10–28) is due to the large amount of missing data in the dataset. The algorithm signals for rescheduling during the 11th day of operation but because of the lack of data it was not done till day 26. Overall, the algorithm with rescheduling switched on gives an average of 81% prediction errors which are less than the error limit, while without rescheduling 65% are less than the error limit. The version with rescheduling on requires 58% more packets than the algorithm without rescheduling. Even so, the number of packets transmitted by SS-2T with rescheduling on is four times less than the default network.

Figure 15 shows the results of performance assessment for the two-tier Slotted-Scheduler and the GUPTA algorithms using the RH LUCE dataset. For the GUPTA algorithm the error limit was set to 0.75°C. For the slotted scheduler the error limit and precision limit were set to 2% and 80%, respectively, and rescheduling was switched off. In both cases the packet limit was 2,150. As can be



(a) Mean imputation error

(b) Box plot of prediction error

(c) Variation in the number of operational nodes with time

Figure 17: Performance of scheduling algorithms, ST LUCE dataset and multihop network.

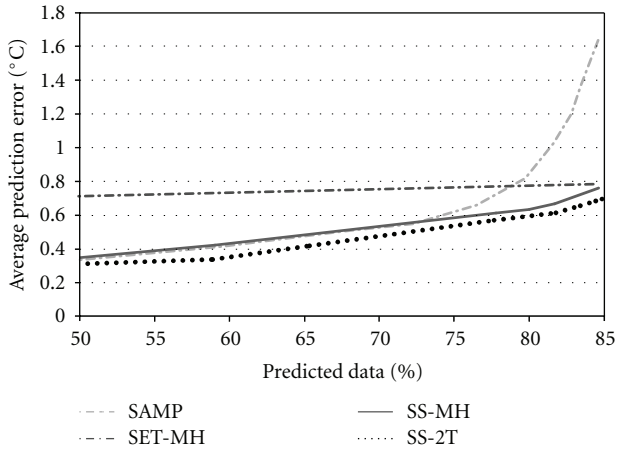


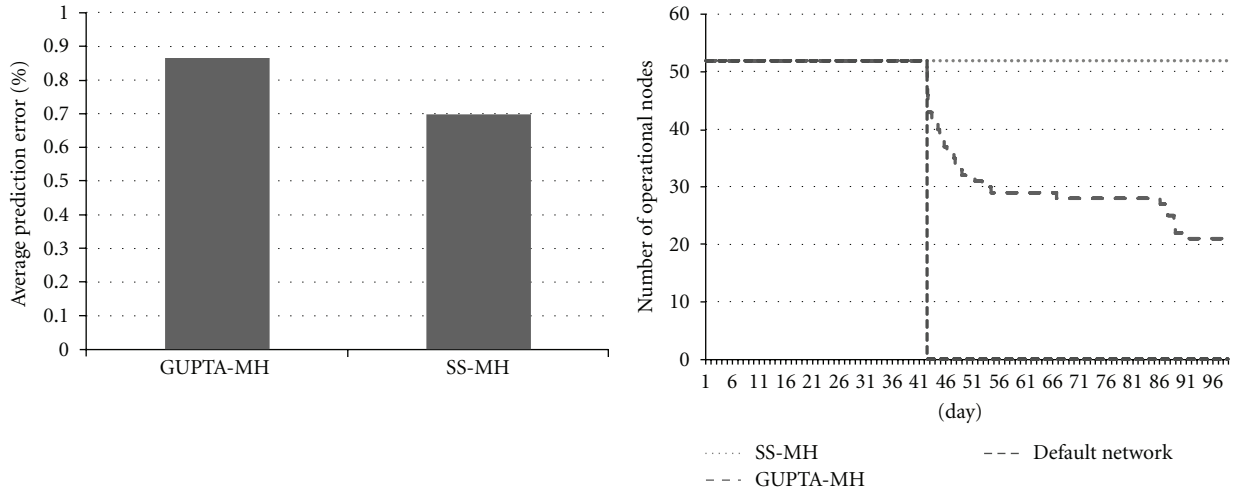
Figure 18: Variation of accuracy with percentage of noncollected data.

seen in Figure 15(a), the Slotted-Scheduler provides greater accuracy: 19% better than GUPTA. Figure 15(b) shows that the nodes running the GUPTA schedule die faster. As shown in Figure 15(c) SS-2T also performs within the 80% precision limit for all time slots during the first week of operation. Table 3 summarizes the results obtained.

Figure 16 compares the performance of SS-2T with rescheduling switched on. With rescheduling switched on, the average percentage of prediction errors below the error limit after day 45 is 80% while for rescheduling off it is 75%. In terms of transmitted packets, during the operational phase, the version with rescheduling transmitted 85% more packets than the version without. The re-scheduled version transmits three times less packets than the default network.

6.2. Multihop Network. Figure 17(a) shows the performance of the multihop algorithms for the ST LUCE dataset. The error limits for the GUPTA and slotted scheduler algorithms are 0.1°C and 0.9°C in 80% of cases, respectively, and rescheduling was switched off. The packet limit is 3,700. The accuracy of the slotted scheduler is 38% better than that of the GUPTA algorithm. In terms of the distribution of prediction error, Figure 17(b) shows that SS-MH performs better than the GUPTA algorithm. Figure 17(c) shows that the Slotted-Scheduler also performs better than the GUPTA algorithm in improving the lifetime in terms of both L_{100} and L_{50}. Table 4 summarizes the performance of the algorithms for the ST LUCE dataset for these precision settings and for one other setting. As can be seen, the Slotted-Scheduler performs within the user-specified error limit.

Figure 18 shows how the accuracy of the subsampling (SAMP), multihop subsetting (SET-MH), multihop slotted



(a) Mean imputation error

(b) Variation in the number of operational nodes with time

Figure 19: Performance of scheduling algorithms, RH LUCE dataset and multihop network.

Table 4: Improvement in lifetime by slotted-scheduler (Mhop) and GUPTA (distributed) with respect to default network.

Data Set	Algorithm	Error limit	Packet limit	Average prediction error	$L_{100\%}$	$L_{50\%}$
ST LUCE	GUPTA-MH	0.1	3700	0.69	0%	30%
ST LUCE	SS-MH	0.9	3700	0.42	145%	145%
ST LUCE	GUPTA-MH	0.5	1700	0.75	0%	233%
ST LUCE	SS-MH	1.8	1700	0.67	444%	444%
RH-LUCE	GUPTA-MH	0.1	3700	0.87	0%	107%
RH-LUCE	SS-MH	1	3700	0.70	133%	133%
RH-LUCE	GUPTA-MH	0.5	1400	1.17	0%	306%
RH-LUCE	SS-MH	3	1400	1.0	553%	553%

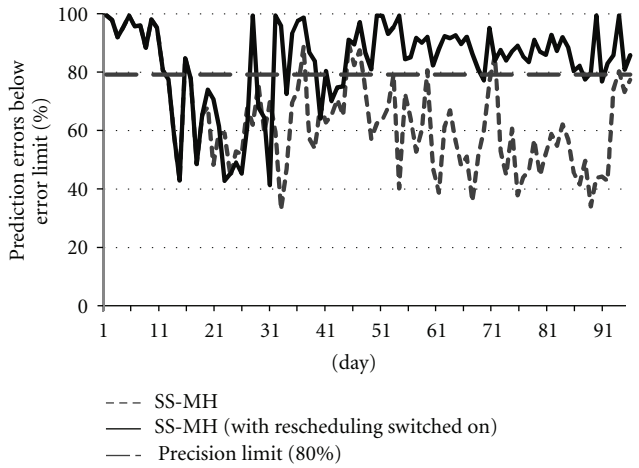


Figure 20: Performance of SS-MH with rescheduling on and off, ST LUCE dataset, multihop network.

scheduler (SS-MH), and two-tier Slotted Schedule (SS-2T, rescheduling off) varies with the percentage of imputed data for the ST LUCE dataset. Again, the slotted scheduler performs better than the subsetting and subsampling algorithm. The performance of the multihop slotted scheduler is similar to that of the two-tier algorithm, even though the subsets are constrained in that they must all provide connectivity to the sink for all nodes.

Figure 20 assesses SS-2T with and without rescheduling. Using rescheduling, the average percentage of prediction errors less than the threshold increases from 65% to 84%. This was achieved at the cost of an 83% increase in the number of packets. As in the two-tier case, even though the algorithm signaled a retrain on day 11, it was unable to perform the retrain for several days due to the amount of missing data.

Figures 19(a) and 19(b) show the performance of the multihop algorithms for the RH LUCE dataset. The error limits are 0.1% for the GUPTA algorithm and 1% in 80% of cases for the slotted scheduler algorithm with rescheduling off. The slotted scheduler outperforms the GUPTA algorithm in terms of both accuracy and lifetime. Accuracy and lifetime summaries are provided in Table 4 for two cases. Figure 21 compares the performance of the multihop subsampling, subsetting and slotted scheduling algorithms with the two-tier

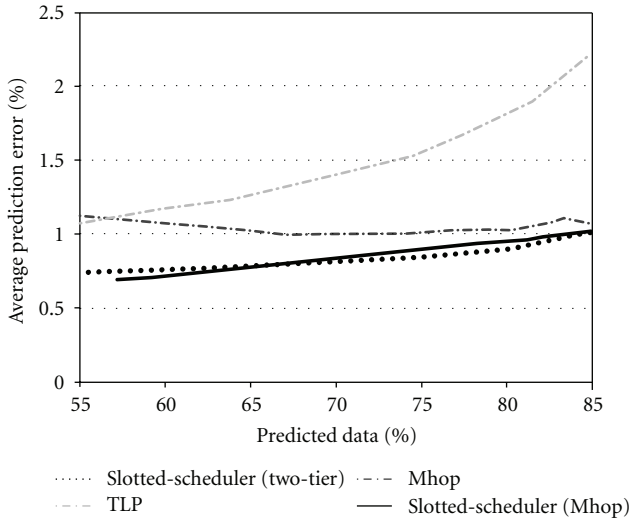


Figure 21: Variation of accuracy with percentage of noncollected data.

slotted scheduler. The previous findings are again confirmed. The findings are similar to the two-tier case.

7. Conclusions

Environmental monitoring applications require nodes to continuously transmit data back to the sink. In this paper we have proposed a method which can use the initial collected data to find internode and temporal correlations within the data. It has been shown that the performance of these internode and temporal models varies across time, between data sets and network densities. Herein a novel adaptive scheduling algorithm has been proposed. The algorithm incorporates novel round-robin subset allocation methods for two-tier and multihop networks. When compared to the previously proposed GUPTA algorithm, the two-tier slotted scheduler provides up to 226% longer lifetime and up to 30% greater imputation accuracy. In a multihop network, the slotted scheduling algorithm improves lifetime by up to 553% and can improve accuracy by up to 38% when compared with the GUPTA algorithm. It has been shown that rescheduling can maintain the performance of the system over a long duration of time at the expense of a small increase in cost in terms of the number of transmitted packets. This adds to the importance of using load balancing to lengthen the time it takes for the first node to die so retraining can be done.

Acknowledgment

This research was funded by Enterprise Ireland under Grant CFTD/07/IT/303.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] V. Shnayder, M. Hempstead, B. R. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 188–200, 2004.

[3] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies. (IEEE INFOCOM '02)*, vol. 3, pp. 1567–1576, 2002.

[4] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems(SenSys '04)*, pp. 95–107, 2004.

[5] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, 2004.

[6] F. X. Li, A. Islam, G. C. Perera, and P. K. Kolli, "Real-time urban bridge health monitoring using a fixed wireless mesh network," in *Proceedings of the IEEE Radio and Wireless Symposium (RWW '10)*, pp. 384–387, January 2010.

[7] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pp. 88–97, 2002.

[8] D. Ingraham, R. Beresford, K. Kaluri, M. Ndoh, and K. Srinivasan, "Wireless sensors: oyster habitat monitoring in the bras d'Or lakes," *Distributed Computing in Sensor Systems*, vol. 3560, pp. 399–400, 2005.

[9] A. Nadamani, P. Basu, and L. Tong, "Extremum tracking in sensor fields with spatio-temporal correlation," in *Proceedings of the IEEE Military Communications Conference (MILCOM '10)*, pp. 1050–1055, 2010.

[10] R. Brown and V. Swail, "Spatial correlation of marine wind-speed observations," *Atmosphere-Ocean*, vol. 26, pp. 524–540, 1988.

[11] G. Dubois, M. Saisana, A. Chaloulakou, and N. Spyrellis, "Spatial correlation analysis of nitrogen dioxide concentrations in the area of Milan, Italy," in *Proceedings of the 1st Biennial Meeting of the International Modeling and Software Society*, pp. 176–183, 2002.

[12] T. Dang, N. Bulusu, and F. W. Rida, "A robust information-driven data compression architecture for irregular wireless sensor networks," *Wireless Sensor Networks*, pp. 133–149, 2007.

[13] M. Welsh and G. Mainland, "Programming sensor networks using abstract regions," in *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, vol. 1, p. 3, USENIX Association, 2004.

[14] A. Boulis, S. Ganeriwal, and M. B. Srivastava, "Aggregation in sensor networks: an energy-accuracy trade-off," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 317–331, 2003.

[15] M. Cardei, My. T. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '05)*, vol. 3, pp. 1976–1984, 2005.

[16] W. Wang, V. Srinivasan, K. C. Chua, and B. Wang, "Energy-efficient coverage for target detection in wireless sensor networks," in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN '07)*, pp. 313–322, 2007.

[17] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in

Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, pp. 32–41, 2002.

[18] H. Gupta, V. Navda, S. Das, and V. Chowdhary, "Efficient gathering of correlated data in sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 1, pp. 1–31, 2008.

[19] S. Yoon and C. Shahabi, "The Clustered AGgregation (CAG) technique leveraging spatial and temporal correlations in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 3, no. 1, Article ID 1210672, p. 3, 2007.

[20] A. Jain and E. Y. Chang, "Adaptive sampling for sensor networks," in *Proceedings of the 1st International Workshop on Data Management for Sensor Networks (DMSN '04)*, pp. 10–16, August 2004.

[21] P. Tillapart, T. Yeophantong, T. Techachaicherdchoo, T. Thumthawatworn, and U. Udomkul, "Adaptive working schedule modeling for wireless sensor networks," in *Proceedings of the IEEE Aerospace Conference*, vol. 9, March 2006.

[22] M. Li, D. Ganesan, and P. Shenoy, "Presto: feedback-driven data management in sensor networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1256–1269, 2009.

[23] S. B. Roy, G. Das, and S. Das, "Computing best coverage path in the presence of obstacles in a sensor field," *Lecture Notes in Computer Science*, vol. 4619, pp. 577–588, 2007.

[24] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate data collection in sensor networks using probabilistic models," in *Proceedings of the International Conference on Data Engineering*, vol. 2006, p. 48, 2006.

[25] L. B. Yann-Ael and B. Gianluca, "Round robin cycle for predictions in wireless sensor networks," in *Proceedings of the 2nd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '05)*, vol. 2005, pp. 253–258, Melbourne, Australia, 2005.

[26] J. C. Lim and C. J. Bleakley, "Extending the lifetime of sensor networks using prediction and scheduling," in *Proceedings of the International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '08)*, pp. 563–568, December 2008.

[27] X. Meng, T. Nandagopal, L. Li, and S. Lu, "Contour maps: monitoring and diagnosis in sensor networks," *Computer Networks*, vol. 50, no. 15, pp. 2820–2838, 2006.

[28] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *Proceedings of the 30th International Conference on Very Large Data Bases*, vol. 30, pp. 588–599, VLDB Endowment, 2004.

[29] D. Apiletti, E. Baralis, and T. Cerquitelli, "Energy-saving models for wireless sensor networks," *Knowledge and Information Systems*, pp. 1–30, 2010.

[30] E. Baralis, T. Cerquitelli, and V. D'Elia, "Modeling a sensor network by means of clustering," in *Proceedings of the 18th International Workshop on Database and Expert Systems Applications (DEXA '07)*, pp. 177–181, September 2007.

[31] Y. Panthachai and P. Keeratiwintakorn, "An energy model for transmission in Telos-based wireless sensor networks," in *Proceedings of the International Joint Conference on Computer Science and Software Engineering (JCSSE '07)*, 2007.

[32] W. Bober and C. Bleakley, "Bailigh: low power cross-layer data gathering protocol for wireless sensor networks," in *Proceedings of the International Conference on Ultra Modern Telecommunications and Workshops (ICUMT '09)*, pp. 1–7, 2009.

[33] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[34] "Wireless distributed sensing system for environmental monitoring, Luce deployment," http://sensorscope.epfl.ch/index.php/Environmental_Data.

[35] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS '04)*, vol. 18, 2004.