RESEARCH ARTICLE

# Adaptive WSN Scheduling for Lifetime Extension in Environmental Monitoring Applications

Jong Chern Lim*†, Chris Bleakley

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

## ABSTRACT

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 spatial and temporal data correlations. 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 multi-hop networks, the proposed algorithm improves on the previously published algorithm by up to 553% and 38% in terms of lifetime and accuracy, respectively.

**KEYWORDS**

Scheduling, Spatial Correlation, Temporal Correlation, Prediction, Wireless Sensor Network

***Correspondence**

Jong Chern Lim, Complex & Adaptive Systems Laboratory, School of Computer Science and Informatics, University College Dublin, Ireland

## 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 micro controller, 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], [4], [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] [7] [8]. WSN measurements of data, such as temperature, humidity, air pressure, wind speed, nitrogen dioxide, and light, often exhibit strong spatial correlation between nodes and strong temporal correlations between different sampling times at the same node [9] [10] [11] [12]. Knowledge of these correlations can be exploited to reduce the number of measurements needed to meet the application-specific sensing accuracy requirements. For example, if outdoor temperature varies more slowly at night than during the day, the sampling rate can be scaled back during the night and increased during the day without unduly affecting accuracy. The missing data can then be estimated (imputed) based on the data actually collected. This saves energy by reducing the amount of data transmitted during the night since nodes can be scheduled to enter sleep modes when they are not needed (see Section 2 for more details).

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

†E-mail: Jong.Lim@ucdconnect.ie

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 (*as most monitoring applications gathers data continuously at the sink, running a preliminary data collection would cost nothing.*). During preliminary data collection, data is collected from all nodes at the full rate. The preliminary data is divided into training and evaluation data sets. The training data is used to build spatial and temporal models of the data relationships. 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 a number of time periods or slots. A different sub-schedule 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, e.g. 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 re-trained and the sub-schedule 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 sub-sampling and round-robin sub-setting scheduling strategies. Variants of the algorithm are proposed for two-tier and multi-hop 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 sub-setting algorithm proposed herein, it is different from coverage based sub-setting algorithms [15] [16] [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 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 multi-hop. Figures 1 and 2 shows an example of a two-tier network and a multi-hop 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 multi-hop 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 multi-hop 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

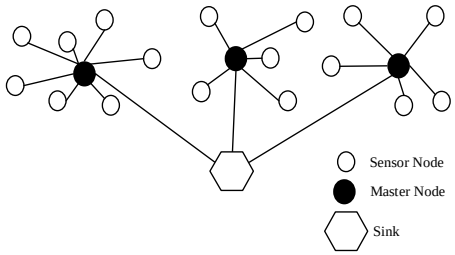



**Figure 1.** Two-tier Network

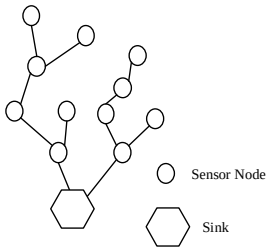



**Figure 2.** Multi-hop Network

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 multi-hop 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 intra-node schedule negotiation and allows for exploitation of multi-hop 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 sub-sampling or sub-setting. Sub-sampling is the process of making measurements less frequently, e.g. a sub-sampling ratio of 2 would increase node sampling periods from 1 minute to 2 minutes. Round robin sub-setting is the process of using only a proportion of the nodes at any one time in a round robin fashion, e.g. a sub-setting ratio of 2 would mean that half the nodes are sampled in even numbered minutes (1, 3, 5,..) and the other half are sampled in odd numbered minutes (0, 2, 4,..). Both of the examples halve the energy consumption of the network but. The level of accuracy in imputing missing data varies depending on how strong the data is temporally or spatially correlated. The algorithm proposed in this work uses both sub-sampling and round robin sub-setting.

A number of publications have dealt with sub-sampling [20] [21] [22]. In all cases, measurements are suppressed, i.e. not transmitted, if there can be accurately predicted based on previous measurements. The suppression can either be *a priori*, before the measurement is taken, or *post prior*, after the measurement is taken. As will be seen, depending on the data set, sometimes sub-setting outperform sub-sampling and sometimes vice verse. Hence the proposed approach supports both sub-setting and sub-sampling.

Several publications have proposed algorithms for sub-setting. These algorithms can be classified according to whether the sub-setting 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] [16] [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 disc shaped sensing radius no longer hold 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 spatial correlations change remodeling/retraining has to be done to maintain a high quality of data gathering service.

A number of methods have been proposed for sub-setting 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 sub-sets. The disadvantage of the distributed approach is that, if subsets are large, initializing and maintaining them requires a significant amount of inter-node communication, as in KEN [24]. As a consequence, Contour Maps and CAG [19] limit the range of sub-sets to one hop. The disadvantage of this is that long distance correlations cannot be exploited. Furthermore this sub-setting 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 multi-hop 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 multi-hop version the GUPTA algorithm is semi-distributed 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 its neighbors.

In order to achieve load balancing for two tier networks two systems have been previously proposed which incorporate round robin sub-setting [25] and [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 sub-setting for multi-hop networks.

Two systems have been previously described which use both sub-setting and sub-sampling - KEN [24] and Contour Maps [27]. Unlike the proposal described herein this 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 sub-setting 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 authors of SeReNe make a brief discussion on possible ways of adapting 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 of the various data similarity based scheduling algorithms that have been proposed is provided in Table I. The algorithm proposed herein is the first to support schedule adaptation and round-robin sub-setting.

## 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 of 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 data of dead nodes can still be spatially imputed, because the node is dead, validation and retraining of the spatial correlation 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 a error of less than $0.5^{o}$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 $P_{lim}$.

$$S_ch : min(N_p) s.t. p(e < E_{lim}) > P_{lim} \quad (1)$$

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. Fig. 3 shows the variation of temperature at three nodes over a day in a real-world dataset. Clearly the rate of change and inter-node data correlations are dependent on the time of day. Thus scheduling algorithm should account of the fact that data correlations drift during the day and, for best performance, should use different sub-schedules at different times of the day. In addition, over long periods of time the temporal and spatial correlations which exist in the data vary. Thus, imputation becomes less accurate. This deterioration in performance should be detected and the models re-trained.

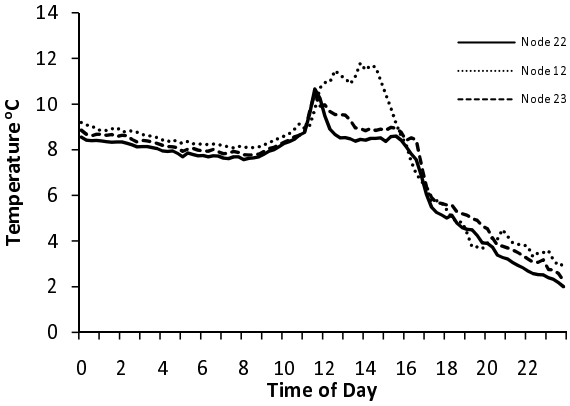



**Figure 3.** Data Relationship of Three Nodes (Temperature-LUCE deployment)

When sub-setting, it is desirable 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 $C_j$, $C_i \cap C_j = \phi$, i.e. 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 non-trivial. In the multi-hop case, the problem is more complex since every disjoint set must provide a representative node to represent each correlated region while also ensuring connectivity between all the nodes in the subset and the sink. For the example, the three disjoint subsets in Figure 4 allow both load balanced sub-setting and continuous connectivity while having each correlated region being represented by a node.

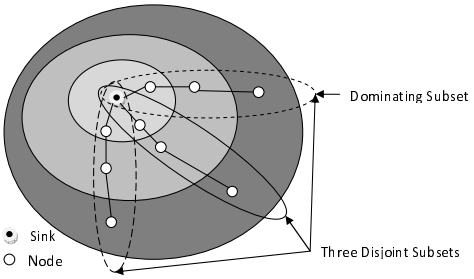



**Figure 4.** Disjoint Sub-setting Example

Figure 5 shows the performance of sub-setting method and sub-sampling with 75% of the data being predicted. Both methods are explained in detail in the

**Table I.** Previous algorithms, main features.

| Algorithm | Reference | Two-tier | Multihop | Centralized /Distributed | Round Robin Sub-setting | Sub-sampling | Adaptive Scheduling |
|---|---|---|---|---|---|---|---|
| CAG | [19] | x | ✓ | Distributed | x | ✓ | x |
| GUPTA | [18] | x | ✓ | Semi-Distributed | x | x | x |
| KEN | [24] | x | ✓ | Distributed | x | ✓ | x |
| SeReNe | [29] | x | ✓ | Centralized | x | x | x |
| RRC | [25] | ✓ | x | Centralized | ✓ | x | x |
| SS-MH/SS-2T | Proposed Method | ✓ | ✓ | Centralized | ✓ | ✓ | ✓ |

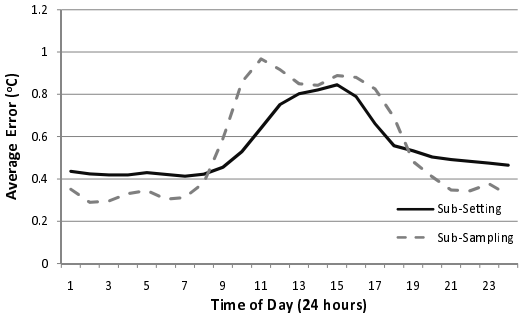



**Figure 5.** Performance of Sub-setting and Sub-sampling Averaged Over 105 Days

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 sub-sampling/sub-setting ratio.

## 4. PROPOSED ALGORITHM

In this section we explain the proposed Slotted-Scheduling algorithm with variants for two-tier (SS-2T) and multi-hop (SS-MH) networks. The following sub-sections provide an overview of the algorithm; explain how schedules are defined; describe how data imputation is performed; explain node to subset allocation for round-robin sub-setting in both two-tier and multi-hop 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 multi-hop 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

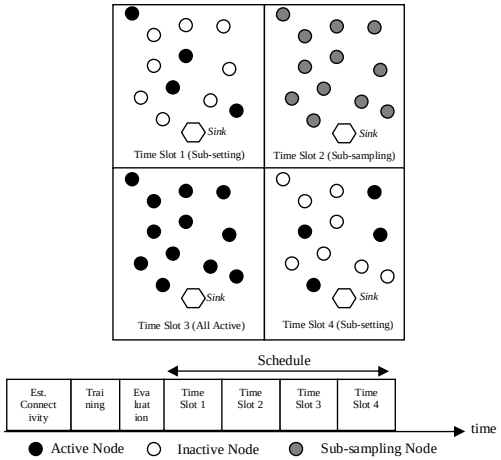



**Figure 6.** Slotted-Scheduler Timeline and Network Activity

data back to the sink. At the sink, the training data is analyzed, on a slot-by-slot basis, to build models for data imputation. The data from the evaluation phase is then used to assess the performance of various candidate scheduling strategies, i.e. various ratios of sub-setting and sub-sampling. The sub-schedule 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/spatial correlation. If changes are detected, the network re-enter the training and evaluation phases in order to update the models and schedule.

Figure 6 illustrates how the Slotted-Scheduling algorithm operates. The figure shows a 4 slot schedule with sub-setting, sub-sampling, full rate collection and sub-setting 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.

### 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, i.e. with no sub-sampling or no-sub-setting applied. The schedule is divided into a number of slots, or time periods, which span the day. A different sub-schedule 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 a.m., slot 1 from 4 a.m. to 8 a.m. and so on. Within each slot, the node sampling sub-schedule is specified as the sampling rate at which the node samples relative to the default collection rate. For example, a sub-sampling 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 sample relative to the start of the operational phase.

A node schedule consists of:

- node id
- eight bits indicating default data collection period (in minutes)
- eight bits indicating slot length
- 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's 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 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 non-collected data. Non-collected data must be imputed based on collected data.

### 4.3. Data Imputation

In the case of sub-sampling, 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 non-collected data as a weighted sum of previous measurements obtained at the same node

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

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 sub-sampling ratio and $p$ is the length of the predictor.

In the case of sub-setting, 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 $s_1, s_2, ..s_L$ 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 than the weighted coefficients are

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

### 4.4. Round-Robin Sub-setting

To achieve load balancing, every node in the network is allocated to a sub-set and the number of nodes per subset is constant. The key to accuracy is in allocating the nodes such that every sub-set 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 sub-setting in two-tier and multi-hop networks.

#### 4.4.1. Two-Tier Networks

In the two-tier case, node to sub-set allocation is achieved by node clustering, followed by sub-set 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.

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

where $\Sigma$ 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.

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

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

$X$ =All sensor nodes
$i = 1$
**while** *X!*=$\emptyset$ **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 $C_q$
  $S_{total} = S_{avg_{C_p}} + S_{avg_{C_q}}$
  $C_{p_{old}}$=$C_p$
  $C_{q_{old}}$=$C_q$
  Swap a random node from $C_p$ and $C_q$
  $S_{total_{new}} = S_{avg_{C_p}} + S_{avg_{C_q}}$
  **if** $S_{total} > S_{total_{new}}$ **then**
    $C_p$=$C_{p_{old}}$
    $C_q$=$C_{q_{old}}$
  **end**
  $count++$
**end**

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

$X$=All sensor nodes
$T_L$=Tranmission 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!*=$\emptyset$ **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 multi-hop round-robin subset allocation

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 optimized 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 sub-setting algorithm is described in Algorithm 1.

Subset allocations and models are generated in this way for a range of sub-setting ratios. The allocations are saved for later evaluation, see subsection 4.5.

#### 4.4.2. Multi-Hop Networks

In the multi-hop cases, allocation of nodes to sub-sets is performed in a different way. This is because, in multi-hop networks, all sub-sets must provide connectivity between all nodes in the subset and the sink. The algorithm works by growing the maximum number of subsets from the sink based on connectivity information and data similarity.

Using a 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. A larger distance criteria will yield a larger number of subsets. The subsets are grown by selecting the nodes according to the following criteria:

- are 1 hop away from a node currently in the subset
- has 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 this maximum number of subsets have 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.

### 4.5. Selecting the Best Schedule

The performance of all possible sub-sampling and sub-setting strategies is assessed for each slot. The sub-sampling or sub-setting sub-schedule 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 non-collected data is imputed and the result compared to the measured data to give the imputation error $e$

$$e(i,t) = abs(x_i(i,t) - x_o(i,t)) \quad (7)$$

The standard deviation of the error $\sigma$ 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 $\sigma_{lim}$. Sub-schedules which lead to error standard deviations in excess of the target $\sigma > \sigma_{lim}$ 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 sub-schedule with the least number of measurements is selected for application to the network. The final schedule is determined by concatenation of the selected sub-schedules. If appropriate, the schedule can be compacted by merging consecutive sub-schedules 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 spatial and temporal 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 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 using equation 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 spatial model, it is first tested using the current samples received from the nodes of the current operational subset $C_i$. Using equation 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 than 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) are 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 on Matlab and tested on two datasets taken from the Lausanne Urban Canopy Experiment (LUCE) [34]. Table II provides a summary of the datasets.

Results were evaluated in terms of mean imputation error (see Eq. 7), percentage of non-collected data, variation of the number of operational nodes with time, and network lifetime. Mean imputation error is the mean error of the imputed non-collected data. The percentage of non-collected 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's data correlation with other nodes change 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 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 number of 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, i.e. all data is collected.

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

**Table II.** 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% | $106600m^2$ |
| ST LUCE (Surface Temperature) | 1/12/2006 | 15 Minutes | 112 | 52 | 7.86% | $106600m^2$ |

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 $newIM$ 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 $B(M \cup s_i, M) = newIM - IM$. The purpose of the benefit function is to maximize the number of inferred nodes thus maximizing the number of sleeping nodes.

**Input**: A sensor network with a correlation graph
**Output**: A correlated dominating set M
BEGIN M = node $s_i$ with largest IM
**while** $B(newM, M > 0$ **do**
Pick $S_i$ for which $B(M \cup S_i, M)$ is maximum
**end**
**Algorithm 3**: GUPTA Centralized Algorithm

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

- The node $s$ has not been mark selected
- The connectivity of the communication subgraph is not affected by the deletion of the node $s$
- 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 nodes which can be predicted by $s$ the higher $p(s)$ will be.

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 were used to build the spatial/temporal model (training) while the subsequent 7 days were used to assess the performance of various sub-setting and sub-sampling ratios (evaluation). It is assumed that the underlying network is able to handle packet loss. In the multi-hop 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 were used for re-scheduling the nodes. The first day is used for a training phase while the second for the evaluation phase. Re-scheduling 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 missing data were deleted from the dataset as this impeded rescheduling.

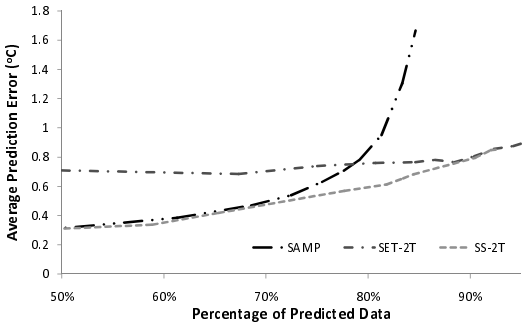



**Figure 7.** Variation in mean imputation error with percentage of non-collected data

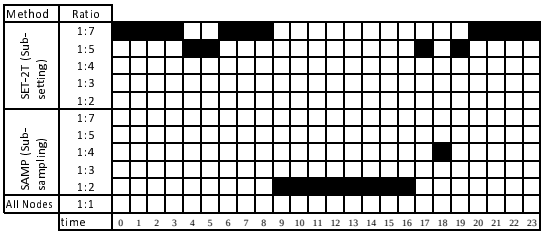
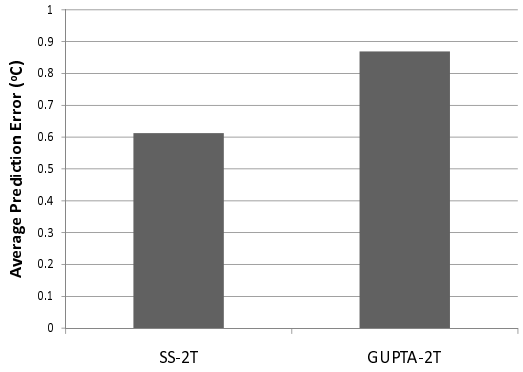


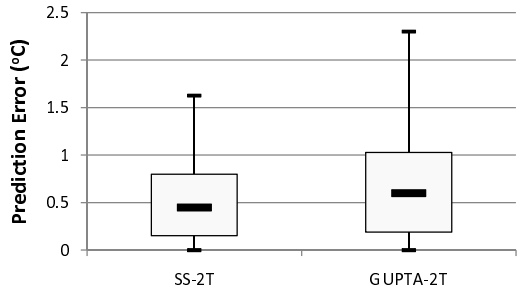

**Figure 8.** Schedule for ST LUSE dataset, target error of $1.4^oC$ in 80% of cases

## 6. RESULTS

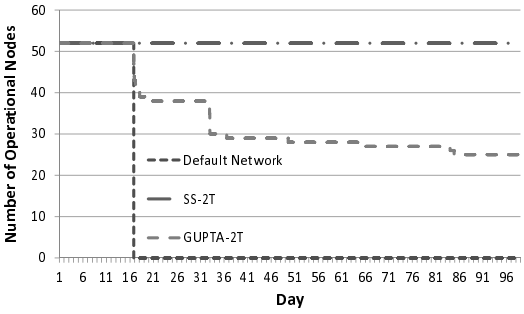
This section is divided into two subsections covering the two-tier and multi-hop cases, respectively.



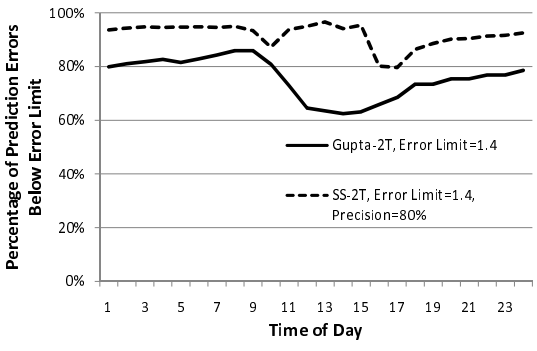

(a) Mean imputation error




(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 9.** Performance of scheduling algorithms, ST LUCE dataset, two-tier network

### 6.1. Two-tier Network

Firstly the performance of the various sub-setting, sub-sampling and imputation methods were assessed using the ST LUCE data set and a two-tier network. Figure 7 shows the variation in mean imputation error with the percentage of imputed data for various methods. Three methods were compared - sub-sampling (SAMP), two-tier sub-setting (SET-2T), and the full Slotted Scheduler algorithm including both sub-setting and sub-sampling (SS-2T). Rescheduling was switched off. For low imputation percentages, sub-sampling performs better than sub-setting. For high imputation percentages, sub-setting performs better than sub-sampling. The proposed Slotted Scheduling algorithm combines the advantages of sub-setting and sub-sampling and performs best in all cases.

Figure 8 shows the schedule created by the Slotted Scheduler for an error limit of $1.4^oC$ in 80% of cases. The figure shows that the choice between sub-setting versus sub-sampling as well as the ratio varies during the day depending on the data statistics. Between the times of 00:00 and 09:00 sub-setting 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), sub-sampling is used with a sampling ratio of 1:2. From 20:00 onwards, the scheduler reverts back to the use of sub-setting.

The GUPTA and Slotted Scheduling algorithms were compared using an error limit of $0.25^o$C and of $1.2^o$C in 80% of cases, respectively. Re-scheduling was switched off. Figure 9(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 9(b). The box plot clearly shows that in terms of the distribution of errors SS-2T performs better as well. Figure 9(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 III shows the in terms of prediction accuracy as well as lifetime SS-2T outperforms GUPTA-2T. Figure 9(d) shows how the percentage of errors that are in excess of the error limit varies across the time slots. 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 10 compares the performance of SS-2T with re-scheduling 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 re-scheduling during the 11th day of operation but because of the lack of data it was not done till day 26. Overall, the algorithm with re-scheduling switched on gives an average of 81% prediction errors which are less than the error limit, while without re-scheduling 65% are less than the error

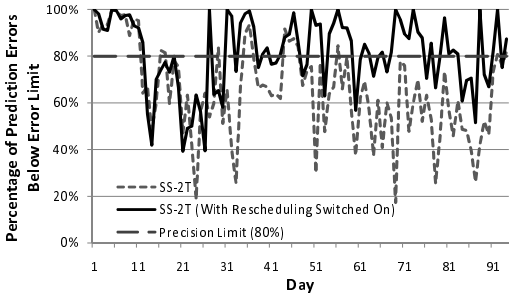



**Figure 10.** Performance of SS-2T with re-scheduling on and off, ST LUCE dataset, two-tier network

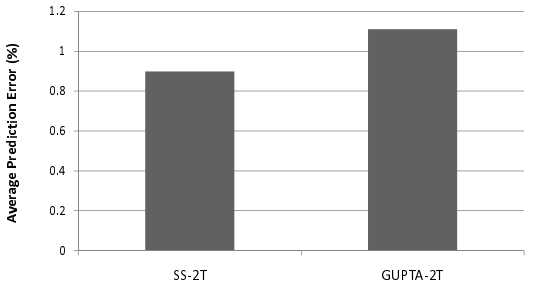
limit. The version with re-scheduling on requires 58% more packets than the algorithm without re-scheduling. Even so, the number of packets transmitted by SS-2T with re-scheduling on is four times less than the default network.

Figure 11(a) 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 $0.75^oC$. For the Slotted-Scheduler the error limit and precision limit were set to 2% and 80% respectively, and re-scheduling was switched off. In both cases the packet limit was 2,150. As can be seen, the Slotted-Scheduler provides greater accuracy: 19% better than GUPTA. Figure 11(b) shows that the nodes running the GUPTA schedule die faster. Table III summarizes the results obtained.

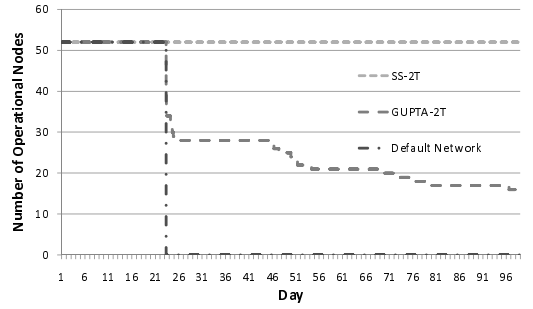
Figure 12 compares the performance of the SS-2T with re-scheduling switched off. With re-scheduling switched on, the average percentage of prediction errors below the error limit after day 45 is 80% while for re-scheduling off it is 75%. In terms of transmitted packets, during the operational phase, the version with re-scheduling transmitted 85% more packets than the version without. The re-scheduled version transmits three times less packets than the default network.

## 6.2. Multi-hop Network

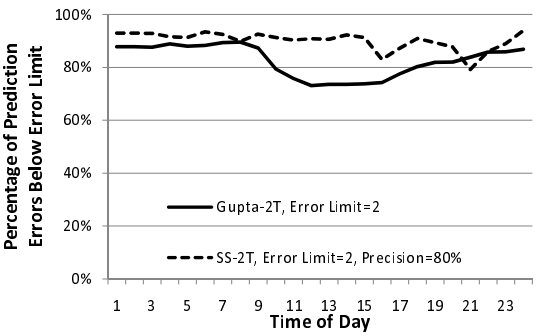
Figure 13(a) shows the performance of the multi-hop algorithms for the ST LUCE dataset. The error limits for the GUPTA and Slotted Scheduler algorithms are $0.1^oC$ and $0.9^oC$ in 80% of cases, respectively, and re-scheduling was switched off. The packet limit is 3,700. The accuracy of the Slotted Scheduler is 38% better than that of the GUPTA algorithm. In terms of the distribution of prediction error figure 13(b) shows that SS-MH performs better than the GUPTA algorithm. The Slotted-Scheduler also performs better than the GUPTA algorithm in improving the lifetime in terms of both $L_{100}$ and $L_{50}$. Table IV 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




(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 11.** Performance of scheduling algorithms, RH LUCE dataset, two-tier network

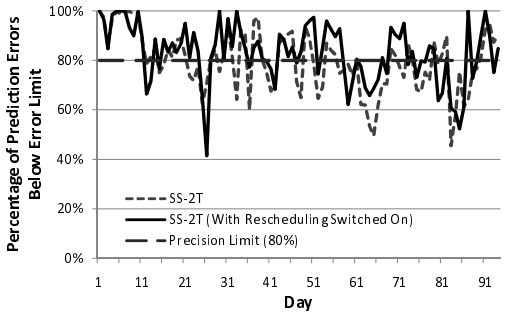
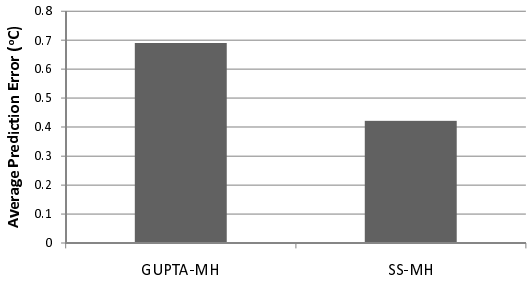
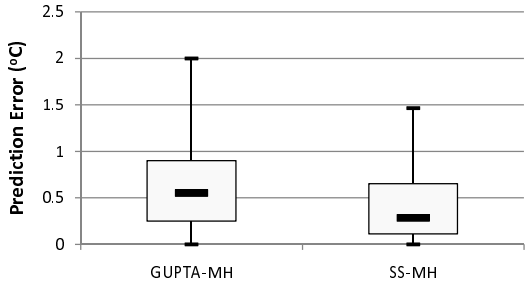


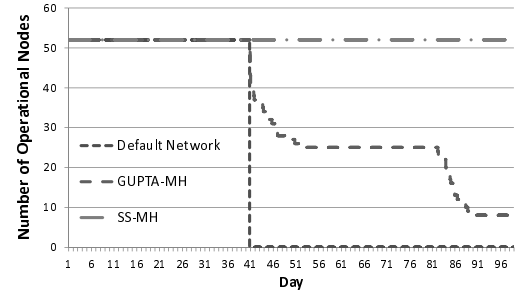

**Figure 12.** Performance of SS-2T with re-scheduling on and off, RH LUCE dataset, two-tier network




(a) Mean imputation error




(b) Box plot of prediction error




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

**Figure 13.** Performance of scheduling algorithms, ST LUCE dataset, multi-hop network

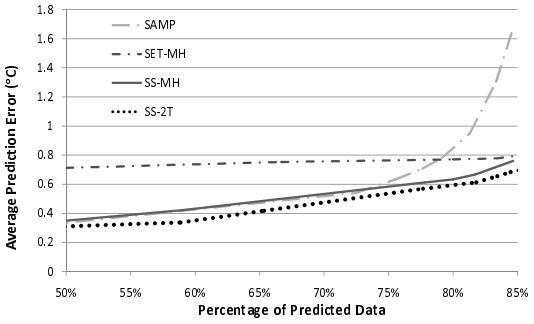
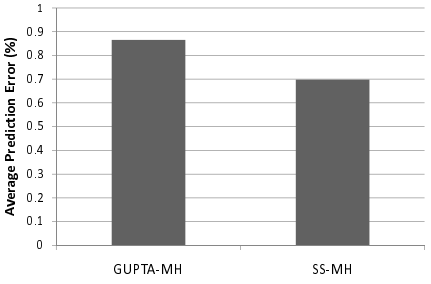
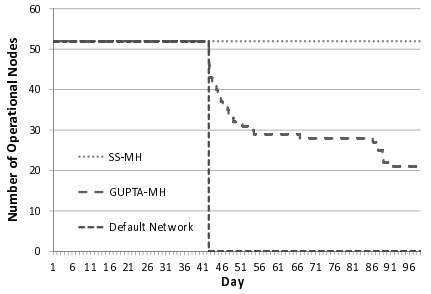



**Figure 14.** Variation of accuracy with percentage of non-collected data

Slotted-Scheduler performs within the user specified error limit.




(a) Mean imputation error




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

**Figure 15.** Performance of scheduling algorithms, RH LUCE dataset, multi-hop network

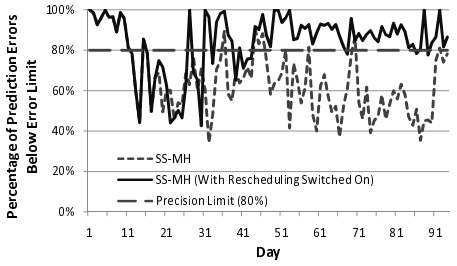



**Figure 16.** Performance of SS-MH with re-scheduling on and off, ST LUCE dataset, multi-hop network

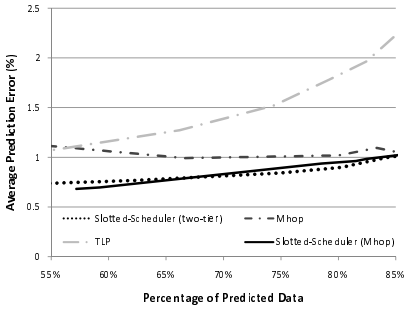



**Figure 17.** Variation of accuracy with percentage of non-collected data

Figure 14 shows how the accuracy of the sub-sampling (SAMP), multi-hop sub-setting (SET-MH), multi-hop Slotted Scheduler (SS-MH) and two-tier Slotted Schedule (SS-2T, re-scheduling off) varies with the percentage of imputed data for the ST LUCE dataset. Again, the performance of Slotted Scheduler performs better than the sub-setting and sub-sampling algorithm. The performance of the multi-hop 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 16 assesses SS-2T with and without re-scheduling. Using re-scheduling, the average percentage of prediction errors less than the threshold increase 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 perform the retrain for several days due to the amount of missing data.

Figures 15(a) and 15(b) show the performance of the multi-hop algorithms for the RH LUCE dataset. The error limits are $0.1\%$ for the GUPTA algorithm and $1\%$C in 80% of cases for the Slotted Scheduler algorithm with re-scheduling off. The Slotted Scheduler outperforms the GUPTA algorithm in terms of both accuracy and lifetime. Accuracy and lifetime summaries are provided in Table IV for two cases. Figure 17 compares the performance of the multi-hop sub-sampling, sub-setting and Slotted Scheduling algorithms with the two-tier Slotted Scheduler. The previous findings are again confirmed. The findings are similar to the two-tier case.

### 6.3. Conclusions

Environmental monitoring applications requires 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 spatial and temporal correlations within the data. It has been shown that the performance of these spatial 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 sub-set allocation methods for two-tier and multi-hop 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 multi-hop 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 re-scheduling can maintain the performance of the system over a long duration of time at a low increase in cost in terms of the number of transmitted packets. Performance results showed by retraining also show the importance of network load balancing, as the moment a node dies it can no longer be retrained.

## 7. ACKNOWLEDGEMENT

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

## REFERENCES

1. Akyildiz I, Su W, Sankarasubramaniam Y, Cayirci E. Wireless sensor networks: a survey. *Computer networks* 2002; **38**(4):393–422.
2. Shnayder V, Hempstead M, Chen B, Allen G, Welsh M. Simulating the power consumption of large-scale sensor network applications. *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ACM New York, NY, USA, 2004; 188–200.
3. Ye W, Heidemann J, Estrin D. An energy-efficient MAC protocol for wireless sensor networks. *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, IEEE, 2002; 1567–1576.
4. Polastre J, Hill J, Culler D. Versatile low power media access for wireless sensor networks. *Proceedings of the 2nd international conference on Embedded networked sensor systems, November*, Citeseer, 2004; 03–05.
5. Ye W, Heidemann J, Estrin D. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on networking* 2004; **12**(3):493–506.
6. Li F, Islam A, Perera G, Kolli P. Real-time urban bridge health monitoring using a fixed wireless mesh network. *Radio and Wireless Symposium (RWS), 2010 IEEE*, IEEE, 2010; 384–387.
7. Mainwaring A, Culler D, Polastre J, Szewczyk R, Anderson J. Wireless sensor networks for habitat monitoring. *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, ACM, 2002; 88–97.
8. Ingraham D, Beresford R, Kaluri K, Ndoh M, Srinivasan K. Wireless Sensors: Oyster Habitat Monitoring in the Bras dOr Lakes. *Distributed Computing in Sensor Systems* 2005; :399–400.
9. Nadamani A, Basu P, Tong L. Extremum Tracking in Sensor Fields with Spatio-temporal Correlation ; .
10. Brown R, Swail V. Spatial correlation of marine wind-speed observations. *Atmosphere-Ocean* 1988; **26**:524–540.
11. Dubois G, Saisana M, Chaloulakou A, Spyrellis N. Spatial correlation analysis of nitrogen dioxide concentrations in the area of Milan, Italy. *Proceedings of the 1st Biennial Meeting of the International Modeling and Software Society*, 2002; 176–183.
12. Dang T, Bulusu N, Feng W. Rida: A robust information-driven data compression architecture for irregular wireless sensor networks. *Wireless Sensor*

**Table III.** Improvement in Lifetime by Slotted-Scheduler (two-tier) and GUPTA (Centralized) With Respect to Default Network (RH LUCE)

| Data Set | Algorithm ($^oC$) | Error Limit | Packet Limit Prediction Error ($^oC$) | Average | $L_{100\%}$ | $L_{50\%}$ |
|---|---|---|---|---|---|---|
| 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% |

**Table IV.** Improvement in Lifetime by Slotted-Scheduler (Mhop) and GUPTA (Distributed) With Respect to Default Network

| Data Set | Algorithm ($^oC$) | Error Limit | Packet Limit Prediction Error ($^oC$) | Average | $L_{100\%}$ | $L_{50\%}$ |
|---|---|---|---|---|---|---|
| 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% |

*Networks* ; :133–149.

13. Welsh M, Mainland G. Programming sensor networks using abstract regions. *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation-Volume 1*, USENIX Association, 2004; 3.
14. Boulis A, Ganeriwal S, Srivastava M. Aggregation in sensor networks: An energy-accuracy trade-off. *Ad hoc networks* 2003; **1**(2-3):317–331.
15. Cardei M, Thai M, Li Y, Wu W. Energy-efficient target coverage in wireless sensor networks. *Proceedings IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, 2005.
16. Wang W, Srinivasan V, Chua K, Wang B. Energy-efficient coverage for target detection in wireless sensor networks. *Proceedings of the 6th international conference on Information processing in sensor networks*, ACM New York, NY, USA, 2007; 313–322.
17. Tian D, Georganas N. A coverage-preserving node scheduling scheme for large wireless sensor networks. *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, ACM, 2002; 32–41.
18. Gupta H, Navda V, Das S, Chowdhary V. Efficient gathering of correlated data in sensor networks. *ACM Trans. Sen. Netw.* 2008; **4**(1):1–31, doi:http://doi.acm.org/10.1145/1325651.1325655.
19. Yoon S, Shahabi C. The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks. *ACM Trans. Sen. Netw.* 2007; **3**(1):3, doi:http://doi.acm.org/10.1145/1210669.1210672.
20. Jain A, Chang E. Adaptive sampling for sensor networks. *ACM International Conference Proceeding Series; Vol. 72*, ACM New York, NY, USA, 2004; 10–16.
21. Tillapart P, Yeophantong T, Techachaicherdchoo T, Thumthawatworn T, Udomkul U. Adaptive working schedule modeling for wireless sensor networks. *2006 IEEE Aerospace Conference*, 2006; 9.
22. Li M, Ganesan D, Shenoy P. Presto: feedback-driven data management in sensor networks. *Networking, IEEE/ACM Transactions on* 2009; **17**(4):1256–1269.
23. Roy B, *et al.*. Computing best coverage path in the presence of obstacles in wireless sensor networks 2007; .
24. Chu D, Deshpande A, Hellerstein J, Hong W. Approximate data collection in sensor networks using probabilistic models. *Proceedings of the 22nd International Conference on Data Engineering*, IEEE Computer Society Washington, DC, USA, 2006; 48.
25. Le Borgne Y, Bontempi G. Round robin cycle for predictions in wireless sensor networks. *2nd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP05), Melbourne, Australia*, 2005.
26. Lim J, Bleakley C. Extending the lifetime of sensor networks using prediction and scheduling. *Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008. International Conference on*, 2008; 563–568.
27. Meng X, Nandagopal T, Li L, Lu S. Contour maps: Monitoring and diagnosis in sensor networks. *Computer Networks* 2006; **50**(15):2820–2838.
28. Deshpande A, Guestrin C, Madden S, Hellerstein J, Hong W. Model-driven data acquisition in sensor

networks. *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, VLDB Endowment, 2004; 588–599.
29. Apiletti D, Baralis E, Cerquitelli T. Energy-saving models for wireless sensor networks. *Knowledge and Information Systems* ; :1–30.
30. Baralis E, Cerquitelli T, D'Elia V. Modeling a Sensor Network by means of Clustering 2007; .
31. Panthachai Y, Keeratiwintakorn P. An energy model for transmission in Telos-based wireless sensor networks. *International joint conference on computer science & software engineering (JCSSE2007)*, 2007.
32. Bober W, Bleakley C. Bailigh: Low power cross-layer data gathering protocol for wireless sensor networks. *Ultra Modern Telecommunications & Workshops, 2009. ICUMT'09. International Conference on*, IEEE, 2009; 1–7.
33. Shi J, Malik J. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence* 2000; **22**(8):888–905.
34. Wireless Distributed Sensing System for Environmental Monitoring, Luce Deployment. *http://sensorscope.epfl.ch/index.php/MainPage* ; .
35. Lu G, Krishnamachari B, Raghavendra C. An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, IEEE, 2004; 224.