Energy-Aware Data Collection in Sensor Networks: A Localized Selective Sampling Approach

Buğra Gedik
College of Computing
Georgia Institute of Technology

Ling Liu
College of Computing
Georgia Institute of Technology

Abstract

One of the most prominent and comprehensive ways of data collection in sensor networks is to periodically extract raw sensor readings. This way of data collection enables complex analysis of data, which may not be possible with in-network aggregation or query processing. However, this flexibility in data analysis comes at the cost of power consumption. In this paper, we introduce *selective sampling* for energy-efficient periodic data collection in sensor networks. The main idea behind selective sampling is to use a dynamically changing subset of nodes as samplers such that the sensor readings of sampler nodes are directly collected, whereas the values of non-sampler nodes are predicted through the use of probabilistic models that are locally and periodically constructed in an in-network manner. Selective sampling can be effectively used to increase the network lifetime while keeping quality of the collected data high, in scenarios where either the spatial density of the network deployment is superfluous relative to the required spatial resolution for data analysis or certain amount of data quality can be traded off in order to decrease the overall power consumption of the network. Our selective sampling approach consists of three main mechanisms. First, *sensing-driven cluster construction* is used to create clusters within the network such that nodes with close sensor readings are assigned to the same clusters. Second, *correlation-based sampler selection and model derivation* is used to determine the sampler nodes and to calculate the parameters of probabilistic models that capture the spatial and temporal correlations among sensor readings. Last, *selective data collection and model-based prediction* is used to minimize the number of messages used to extract data from the network. A unique feature of our selective sampling mechanisms is the use of localized schemes, as opposed to the protocols requiring global information, to select and dynamically refine the subset of sensor nodes serving as samplers and the model-based value prediction for non-sampler nodes. Such runtime adaptations create a data collection schedule which is self-optimizing in response to changes in energy levels of nodes and environmental dynamics.

Keywords: C.2.7.c Sensor networks, C.2.0.b Data communications, H.2.1.a Data models

1 Introduction

Advances in wireless network technologies, low-power processor and chip design, and micro electromechanical systems have facilitated the proliferation of small, low cost, low power sensor devices that enable seamless integration of the physical world with pervasive networks [17]. The prominent features of such sensor devices are their ability to perform computation, wireless communication, and environmental sensing. On the bright side, the continued price drop in low power sensor devices and their decentralized and unattended nature of operation make sensor networks an attractive tool for extracting and gathering data by sensing real-world phenomena from the physical environment. Environmental monitoring applications are expected to benefit enormously from these developments, as evidenced by recent sensor network deployments supporting such applications [33, 6].

On the downside, the large and growing number of networked sensors and their unattended deployment present a number of unique system design challenges, different from those posed by existing computer networks: (1) *Sensors are power-constrained.* A major limitation of sensor devices is their limited battery life. Wireless communication is a major source of energy consumption, where sensing can also play an important role [15] depending on the particular type of sensing performed (ex. solar radiation sensors [41]). On the other hand, computation is relatively less energy consuming. Motes [23] developed at UC Berkeley and manufactured by Crossbow Inc. [14] are good examples of this type of sensor nodes. (2) *Sensor networks must deal with high system dynamics.* Sensor devices and sensor networks experience a wide range of dynamics, including spatial and temporal change trends in the sensed values that contribute to environmental dynamics, changes in user demands that contribute to task dynamics as to what is being sensed and what is considered interesting changes [18], and changes in the energy levels of the sensor nodes, their location or connectivity that contribute to network dynamics. One of the main objectives in configuring networks of sensors for large scale data collection is to achieve longer lifetimes for sensor network deployments by keeping energy consumption at minimum, while maintaining sufficiently high quality and resolution of the collected data to enable meaningful analysis. These configurations should be periodically re-adjusted to adapt to the various changes resulting from high system dynamics.

Data Collection in Sensor Networks – We can broadly divide data collection, a major functionality supported by sensor networks, into two categories. In *event based* data collection, the sensors are responsible for detecting and reporting (to a base node) events, such as spotting moving targets [27]. Event based data collection is less demanding in terms of the amount of wireless communication, since local filtering is performed at the sensor nodes, and only events are propagated to the base node. In certain applications, the sensors may need to collaborate in order to detect events. Detecting complex events may necessitate non-trivial distributed

algorithms [29] that require involvement of multiple sensor nodes. An inherent downside of this kind of data collection is the impossibility of performing in-depth analysis on the raw sensor readings, since they are not extracted from the network.

In *periodic data collection*, periodic updates are sent to the base node from the sensor network, based on the most recent information sensed from the environment. We further classify this approach into two. In *query based* data collection, long standing queries (also called continuous queries [30]) are used to express user or application specific information interests and these queries are installed "inside" the network. Most of the schemes following this approach [31, 32] support aggregate queries, such as minimum, average, and maximum. These types of queries result in periodically generating an aggregate of the recent samples of all nodes. Although aggregation lends itself to simple distributed implementations that enable complete in-network processing of queries, it falls short in supporting holistic aggregates [31] over sensor samples, such as quantiles. Similar to the case of event based data collection, the raw data is not extracted from the network and complex data analysis that requires integration of samples from various nodes at various times, cannot be performed with in-network aggregation.

The most comprehensive way of data collection is to extract raw samples from the network through periodic reporting of each sampled value from every sensor node. This scheme enables arbitrary data analysis at a sensor stream processing center once the data is collected. Such increased flexibility in data analysis comes at the cost of high energy consumption due to excessive communication and consequently decreases the network lifetime. One way of tackling this problem is to use distributed data compression to reduce the total size of the data transmitted on the wireless channel. However, such approaches may require to gather samples belonging to different time intervals before performing compression on them [3]. This may introduce delays, undesirable for real-time applications. As shown in [3], compression techniques that typically trade-off accuracy and delay can cut down the communication cost, thus reduce the energy consumption rate and increase the network lifetime. In this paper, we develop an alternative approach based on *selective sampling*. The main idea behind selective sampling is to use a carefully selected dynamically changing subset of nodes to sample and to predict the values of the rest of the nodes using probabilistic models. Such models are constructed by exploiting both spatial and temporal correlations existent in sample readings of sensor nodes. There are two major scenarios that can highly benefit from this approach. First, in many sensor network applications, node density of the deployment is selected to result in a spatial resolution higher than the required, mainly because of the short lifespan of the sensor nodes [39], or due to the lack of knowledge about the nature of the phenomenon of interest. As a result, selective sampling can effectively reduce the number of nodes used to sample data, decrease the energy consumption rate of the network, and thus can increase the overall network lifetime. Second and more importantly, there is an inherent trade-off between the accuracy of the collected

data and the network lifetime. If the application at hand can tolerate certain levels of error, then selective sampling can be effectively used to save energy by decreasing the quality of received data within acceptable bounds. Such tradeoff is especially useful when the energy left in the network is low and the energy consumption rate is high. A key challenge is to design effective mechanisms that can increase the lifetime of the network while keeping the accuracy of the collected data at satisfactory levels.

Figure 1 illustrates this trade off graphically. Initially, perfect accuracy is sustained with high rate of energy consumption. Later, selective sampling is used to decrease the rate of energy consumption, while introducing some reduction in data quality, to obtain an increased network lifetime. Note that, it is also possible to use different degrees of selective sampling, depending on the desired energy/quality trade-off.

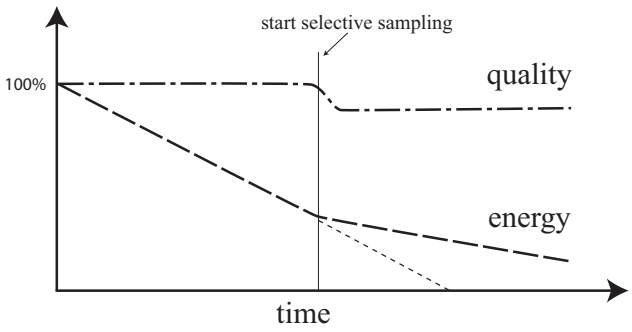

Figure 1: Illustration of energy-quality trade-off

Another key challenge in designing an energy efficient selective sampling architecture is to empower the system with the ability to respond to high network dynamics. Concretely, the selective sampling approach should support large number of unattended autonomous nodes and should equip the energy-efficient data collection algorithms with self-configuring and self-optimizing capabilities by enabling run-time adaptation to re-select the subset of nodes to sample and to re-construct the correlation-based probabilistic models to enhance the quality of value prediction of non-sampler nodes.

Contributions and Scope of the Paper

With the above challenges in mind, we identify a number of concrete design principles in designing an effective selective sampling architecture that can respond to changes in energy levels at nodes and network dynamics. First, we need to organize the network into coordination groups such that good probabilistic models can be locally constructed to closely capture the spatial correlations of sensor readings amongst the nodes within each group. Second, we need to utilize the constructed models to find and select the sampler nodes whose sensor readings can provide high accuracy for the prediction to be performed for the non-sampler nodes. Third, but not the least, we need to perform periodic reassignments in order to balance power consumption of the nodes and adapt to possibly changing correlations between sensor readings.

Our selective sampling architecture consists of a three-phase framework and a set of localized algorithms for generating and executing energy-aware data collection schedules. First, we develop *Sensing-driven Cluster Construction* algorithm to group together the nodes such that the ones that are close to each other in terms of their sensor readings (thus the name *sensing-driven*) as well as network hops are put into the same clusters.

This is aimed at building a network organization that facilitates local coordination for performing selective sampling and is designed to improve the prediction quality via its sensing-driven nature. Second, we develop *Correlation-based Sampler Selection and Model Derivation* algorithms to partition the nodes within each cluster into a set of subclusters to assist the selection of a set of sampler nodes and to construct one probabilistic model for each subcluster. We address the issues of high prediction accuracy and balanced power consumption by enabling periodic re-configuration of node clusters and periodic re-selection of sampler nodes and re-construction of correlation-based probabilistic models. This allows our selective sampling approach to adapt to possibly changing correlations between sensor readings and balance power consumption of nodes in response to environment and task dynamics. In the third phase, we generate and execute the data collection schedule to collect data from the sensor network in an energy-efficient manner by developing the *Selective Data Collection and Model-based Prediction* algorithms, aiming at keeping the wireless communication at minimum. This enables us to strike a good balance between network lifetime and data quality.

2 System Model and Overview

We describe the system model and introduce the basic concepts through an overview of the selective sampling architecture and a brief discussion on the set of algorithms employed. For reference convenience, we list the set of basic notations used in the paper in Tables 1, 2, 3, and 4. Each table lists the set of notations introduced in its associated section.

2.1 Network Architecture

We design our selective sampling based data collection system using a three-layer network architecture. The first and basic layer is the wireless network formed by N sensor nodes and a *data collection tree* constructed on top of the network. We denote a node in the network by p_i, where $i \in \{1, \ldots, N\}$. Each node is assumed to be able to communicate only with its neighbors, that is, the nodes within its communication range. The set of neighbor nodes of node p_i is denoted by $nbr(p_i)$. The neighbor relationship is assumed to be symmetric. The nodes that can communicate with each other form a *connectivity graph*. Figure 2 depicts a segment from a network of hundred sensor nodes. The edges of the connectivity graph are shown with light blue lines (light gray in grayscale). Sensor nodes use a data collection tree for the purpose of propagating their sensed values to a base node. The base node is also the root of the data collection tree. This tree is formed in response to a data collection request, which starts the data collection process. In Figure 2, base node is the shaded one labeled as "56". Every node in the data collection tree, except the root, has a parent node and every non-leaf node has a set of children nodes. The edges of the data collection tree are shown in red color (dark gray in grayscale) in Figure 2. The data collection tree can be easily build in a distributed manner, for instance, by circulating a tree formation message originated from the base node and making use of a min-hop parent

selection policy [3], or similar algorithms used for in-network aggregation [32, 31].

The second layer of the architecture consists of node clusters, which partition the sensor network into disjoint regions. Each node in the network belongs to a cluster and each cluster elects a node within the cluster to be the cluster head, and creates a *cluster-connection tree* with the cluster head as its root node to establish the communication between nodes and their cluster head (see Section 3.2 for further detail). We associate each node p_i with a cluster head indicator h_i, $i \in \{1, \ldots, N\}$, to denote the cluster head node of the cluster that node p_i belongs to. The set of cluster head nodes are denoted by H, and is defined formally as $H = \{p_i |\ h_i = p_i\}$. Note that $h_i = p_i$ implies that p_i is a cluster head node (of cluster i). A cluster with p_i as its head node is denoted by C_i and is defined as the set of nodes that belong to it, including its cluster head node p_i. Formally, $C_i = \{p_j |\ h_j = p_i\}$. Given a node p_j has p_i as its cluster head ($h_j = p_i$), we say p_j is in C_i ($p_j \in C_i$). A cluster is illustrated on the upper left corner of Figure 2 with a closed line covering the nodes that belong to the cluster. The cluster head node is drawn in bold and is labeled as "12". An example cluster-connection tree is shown in the figure, where its edges are drawn in dark blue (using dashed lines).

Notation	Meaning
N	Total number of nodes in the network
p_i	ith node in the network
$nbr(p_i)$	Neighbors of node p_i in the connectivity graph
$e_i(t)$	Energy left at node p_i at time t
h_i	Cluster head node of the cluster that node p_i belongs to
H	Set of cluster head nodes in the network
C_i	Set of nodes in the cluster with head node p_i
G_i	Set of subclusters in cluster C_i, where $G_i(j)$ is the set of nodes in the jth subcluster in G_i
K_i	Number of subclusters in G_i, also denoted as $\|G_i\|$
S_i	Data collection schedule for cluster C_i, where $S_i[p_j]$ is the status (sampler/non-sampler) of node p_j in S_i

Table 1: Notations for network architecture

The third layer of our architecture is built on top of the node clusters in the network, by further partitioning each node cluster into a set of *subclusters*. Each node in the network belongs to a subcluster. The set of subclusters in C_i is denoted by G_i, where the number of subclusters in C_i is denoted by K_i where $K_i = |G_i|$. A subcluster within G_i is denoted by $G_i(j), j \in \{1, \ldots, K_i\}$, and is defined as the set of nodes that belong to the jth subcluster in G_i. Given a node cluster C_i, only the head node p_i of this cluster knows all its subclusters ($G_i(j), j \in \{1, \ldots, K_i\}$). Thus the subcluster information is local to the cluster head node p_i and is transparent to other nodes within the cluster C_i. In Figure 2, we show four subclusters for the node cluster with node "12" as its cluster head and these subclusters are circled with closed dashed lines. A key feature of our selective sampling approach is that not all the nodes in the network need to sample and send the sampled values (sensor readings) to the base node via the data collection tree. One of the design ideas is to partition the node cluster in such a way that we can elect a few nodes within each subcluster as the sampling nodes and create a probabilistic model to predict the values of other nodes within

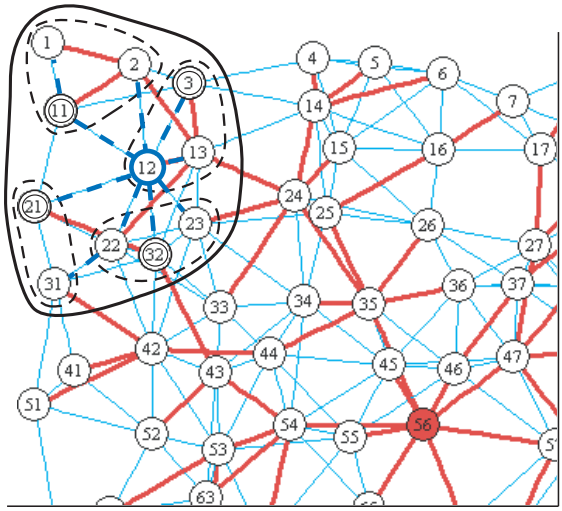

Figure 2: System Architecture

this subcluster. From now on, we refer to the nodes that do sampling as *sampler* nodes. In Figure 2, we show sampler nodes with double circled lines (i.e., nodes labeled "3", "11", "21", and "32"). For each cluster C_i, there exists a data collection schedule S_i, which defines the nodes that are samplers in this node cluster. We use the Boolean predicate denoted by $S_i[p_j]$ as an indicator that defines whether node $p_j \in C_i$ is a sampler or not. We use the [] notation whenever the indexing is by nodes.

2.2 Selective Sampling Overview

We give an overview of the three main mechanisms that form the crux of our selective sampling approach to data collection. A detailed description of each mechanism is provided in the subsequent sections.

The first mechanism is to construct clusters within the network. This is achieved by the *sensing-driven cluster construction* algorithm, that is executed periodically at every τ_c seconds, in order to perform cluster refinement by incorporating changes in the energy level distribution and the sensing behavior changes of the nodes. We call τ_c the *clustering period*. The node clustering algorithm performs two main tasks $-$ cluster head selection and cluster formation. The cluster head selection component is responsible for defining the guidelines on how to choose certain number of nodes in the network to serve as cluster heads. An important design criterion for cluster head selection is to make sure that on the long run the job of being a cluster head is evenly distributed among all the nodes in the network to avoid burning out the battery life of certain sensor nodes too earlier. The cluster formation component is in charge of constructing clusters according to two metrics. First, nodes that are similar to each other in terms of their sampled values (sensor readings) in the past should be clustered into one group. Second, nodes that are clustered together should be close to each other within certain network hops. The first metric is based on value similarity of sensor readings, which is a distinguishing feature compared to naive minimum-hop based cluster formation where a node joins the cluster that has the closest cluster head node in terms of network hops.

The second mechanism is to create the subclusters for each of the node clusters. The goal of further dividing the node clusters into subclusters is to facilitate the selection of the nodes to serve as samplers and the generation of the probabilistic models for value predication of non-sampler nodes. This is achieved by the *correlation-based sampler selection and model derivation* algorithm that is executed periodically at every τ_u seconds. τ_u is called the *schedule update period* and is typically defined as a multiple of τ_u. Concretely, given a node cluster, the cluster head node carries out the sampler selection and model derivation task locally in three steps. In the first step, the cluster head node uses historical data from nodes in its cluster to capture the spatial and temporal correlations in sensor readings and calculate the subclusters so that the nodes whose sample values are highly correlated are put into the same subclusters. In the second step, these subclusters are used to select a set of sampler nodes such that there is at least one sampler node selected from each subcluster.

This selection of samplers forms the sampling schedule for the cluster. We introduce a system-wide parameter $\sigma \in (0, 1]$ to define the average fraction of nodes that should be used as samplers. σ is called the *sampling fraction*. Once the sampler nodes are determined, only these nodes collect sample readings and the values of the non-sampler nodes will be predicted at the processing center (or the base node) using a probabilistic model that is constructed for each subcluster. Thus, the third step here is to construct and report a probabilistic model for each subcluster within the network based on the historical readings of all nodes in the subcluster. We introduce a system-supplied parameter β, which defines the average size of the subclusters. β is called the *subcluster granularity* and its setting influences the size and number of the subclusters used in the network.

The third mechanism, is to collect the sampled values from the network and to perform the prediction after the samples are received. This is achieved by the *selective data collection and model-based prediction* algorithm. The selective data collection component works in two steps: (1) Each sampler node samples its reading every τ_d seconds, called the *desired sampling period*. τ_d sets the temporal resolution of the data collection. (2) To empower our selective sampling architecture with self-adaptation, we also need to periodically sample sensor readings from all nodes in the network. Concretely, at every τ_f seconds (τ_f is a multiple of τ_d) all nodes perform sampling. These samples are collected (through the use of cluster-connection trees) and used by the cluster head nodes, aiming at incorporating newly established correlations among sensor readings and network dynamics into decision making process of correlation-based sampler selection and model derivation. τ_f is a system-supplied parameter, called the *forced sampling period*. The model-based predication component is responsible for estimating values of non-sampler nodes within each subcluster using readings of the sampler nodes and the parameters of the probabilistic model constructed for that subcluster.

3 Sensing-driven Cluster Construction

The goal of sensing-driven cluster construction is to form a network organization that can facilitate selective sampling through localized algorithms, while achieving the global objectives of energy-awareness and high quality data collection. In particular, clusters help perform operations such as sampler selection and model derivation in a localized manner. By emphasizing on sensing-driven clustering, it also helps to derive better prediction models to increase the prediction quality. The sensing-driven clustering algorithm, executed periodically at every τ_c seconds, performs two main tasks $-$ cluster head selection and cluster formation.

3.1 Cluster Head Selection

During the cluster head selection phase, nodes decide whether they should take the role of a cluster head or not. Concretely, every node is initialized not to be a cluster head and does not have an associated cluster in the beginning of a cluster head selection phase. A node p_i first calculates a value called *head selection probability*,

denoted by s_i. This probability is calculated based on two factors. The first one is a system wide parameter called *cluster count factor*, denoted by f_c. It is a value in the range (0,1] and defines the average fraction of nodes that will be selected as cluster heads. In other words, $f_c * N$ number of nodes will be selected as cluster heads on the average, with an average cluster size of $1/f_c$. The factors that can affect the decision on the number of clusters and thus the setting of f_c include the size and density of the network. The second factor involved in the setting of s_i is the *relative energy level* of the node. We denote the energy available at node p_i at time t as $e_i(t)$.

Notation	Meaning
s_i	Head selection probability
r_i	Round counter of node p_i used for clustering
TTL	Max. number of hops a cluster formation message can travel
μ_i	Mean of the sensor readings node p_i has sampled
T_i	Smallest hop distances from cluster heads in proximity of p_i, as known to p_i during cluster formation
V_i	Means of readings from cluster heads in proximity of p_i, as known to p_i during cluster formation
Z_i	Attraction scores for cluster heads in proximity of p_i, where $Z_i[p_j]$ is the attraction score for node $p_j \in H$
f_c	Cluster count factor
α	Data importance factor
τ_c	Clustering period

Table 2: Notations for sensing-driven clustering

The relative energy level is calculated by comparing the energy available at node p_i with the average energy available at the nodes within its one hop neighborhood. The value of the head selection probability is then calculated by multiplying the cluster count factor with the relative energy level. Formally, $s_i = f_c * \frac{e_i(t)*(|nbr(p_i)|+1)}{e_i(t)+\sum_{p_j \in nbr(p_i)} e_j(t)}$. This enables us to favor nodes with higher energy levels for cluster head selection. Once s_i is calculated, node p_i is chosen as a cluster head with probability s_i. If selected as a cluster head, p_i initializes a number of states before starting to circulate cluster formation messages to begin the cluster formation process (described in the next subsection). Concretely, p_i sets h_i to p_i indicating that it now belongs to the cluster with head p_i (itself) and also increments its *round counter*, denoted by r_i to note that a new cluster has been selected for the new clustering round. If p_i is not selected as a cluster head, it waits for some time to receive cluster formation messages from other nodes. If no such message is received, it repeats the whole process starting from the s_i calculation. Considering most realistic scenarios governing energy values available at nodes and practical settings of f_c (< 0.2), this process results in selecting approximately $f_c * N$ number of nodes as cluster heads. The pseudo code is given in the CLUSTINIT procedure of Alg. 1.

3.2 Cluster Formation

The cluster formation phase starts right after the cluster head selection phase. It organizes the network of sensors into node clusters in two major steps: *message circulation* and *cluster engagement*.

Message circulation step involves the circulation of cluster formation messages within the network. These messages are originated at cluster head nodes. Once a node p_i is chosen to be a cluster head, it prepares a message m to be circulated within a bounded number of hops, and structures the message m as follows. It sets $m.org$ to its node identifier p_i. This field represents the originator of the cluster formation message. It sets $m.ttl$ to TTL, where TTL is a system-wide parameter that defines the maximum number of hops this

message can travel within the network. This field indicates the number of remaining hops the message can travel. It sets $m.rnd$ to its round counter r_i. It sets $m.src$ to p_i, indicating the sender of the message. Finally, it sets $m.dmu$ to μ_i. Here, μ_i denotes the mean of the sensor readings node p_i has sampled during the time period preceding this round (r_i) of cluster formation. The message m is then sent to all neighbors of node p_i.

Upon reception of a message m at a node p_i, we first compare the rnd field of the message to p_i's current round counter r_i. If $m.rnd$ is smaller than r_i, we discard the message, since it is likely a delayed message in some earlier clustering round and should be disregarded. If $m.rnd$ is larger than r_i, then this is the first cluster formation message p_i has received for the new round. As a result, we increment r_i to indicate that node p_i is now part of the current round. Moreover, we initialize two data structures, denoted by T_i and V_i. Both are initially empty. $T_i[p_j]$ stores the shortest known hop count from a cluster head node p_j to node p_i, if a cluster formation message is received from p_j. $V_i[p_j]$ stores dmu field of the cluster formation messages that originated from node p_j and reached node p_i. Once the processing of the rnd field of the message is over, we calculate the number of hops this message traveled, by investigating the ttl field, which yields the value $1 + TTL - m.ttl$. If $T_i[m.org]$ is not empty (meaning this is not the first message we received in this round that originated from node $m.org$) and $T[m.org]$ is smaller than or equal to the number of hops the current message has traveled, we discard the message. Otherwise, we set $T[m.org]$ to $1 + TTL - m.ttl$ and $V_i[m.org]$ to $m.dmu$. Once T_i and V_i are updated with the new information, we modify and forward the message to all neighbors of p_i, except the node specified by src field of the message. The modification on the message involves decrementing the ttl field and setting the src field to p_i. The pseudo code for the message circulation phase is given within the CLUSTINIT and RECEIVEMSG procedures in Alg. 1.

Alg. 1: Sensing-driven cluster construction

```
CLUSTINIT(p_i)
(1)  h_i ← nil
(2)  while h_i = nil
(3)      t ← Current time
(4)      s_i ← f_c * e_i(t)*(|nbr(p_i)|+1) / (e_i(t)+Σ_{p_j∈nbr(p_i)} e_j(t))
(5)      if rand(0,1) < s_i
(6)          h_i ← p_i
(7)          r_i ← r_i + 1  /* r_i ← 0 during system init */
(8)          m.org ← p_i, m.ttl ← TTL
(9)          m.rnd ← r_i, msg.src ← p_i
(10)         m.dmu ← μ_i
(11)         foreach p_j ∈ nbr(p_i)
(12)             SENDMSG(p_j, m)
(13)     else if a cluster formation message received
(14)         return

RECEIVEMSG(p_i, m)
(1)  if m.rnd > r_i
(2)      r_i ← m.rnd
(3)      T_i ← ∅, V_i ← ∅
(4)  else if m.rnd < r_i
(5)      return
(6)  a ← 1 + TTL − m.ttl
(7)  if T_i[m.org] ≠ ∅ and T_i[m.org] < a
(8)      return
(9)  T_i[m.org] ← a, V_i[m.org] ← m.dmu
(10) m.ttl ← m.ttl − 1, m.src ← p_i
(11) foreach p_j ∈ nbr(p_i)\msg.src
(12)     SENDMSG(p_j, m)

PICKCLUSTERS(p_i)
(1)  y ← 1/N(μ_i| μ_i, σ_i)
(2)  foreach j, T_i[p_j] ≠ ∅
(3)      a ← 1 − T_i[p_j]/TTL  /* hop distance factor */
(4)      b ← N(V_i[p_j]| μ_i, σ_i) * y  /* data distance factor */
(5)      Z_i[p_j] ← a + α * b
(6)  h_i ← argmax_{p_j}(Z_i[p_j])
```

Cluster engagement step involves making a decision about which cluster to join, once hop distance and mean sample value information are collected. Concretely, a node p_i that is not a cluster head, performs the following

procedure to determine its cluster. For each cluster head node from which it has received a cluster formation message in this round (i.e. $\{p_j|\ T_i[p_j] \neq \emptyset\}$), it calculates an *attraction score*, denoted by $Z_i[p_j]$, for the cluster head p_j. Then it joins the cluster head with the highest attraction score, i.e., it sets h_i to $argmax_{p_j}(Z_i[p_j])$. The calculation of the attraction score $Z_i[p_j]$ involves two factors. The first factor is called the *hop distance factor* and is calculated as $1 - T_i[p_j]/TTL$. It takes its minimum value 0 when p_i is TTL hops away from p_j and its maximum value $1 - 1/TTL$ when p_i is one hop away from p_j. The second factor is called the *data distance factor* and it is calculated as $\mathcal{N}(V_i[p_j]|\ \mu_i, \varsigma_i^2)/\mathcal{N}(\mu_i|\ \mu_i, \varsigma_i^2)$. Here, $\mathcal{N}$ represents the Normal distribution and ς_i^2 is a locally estimated variance of the sampled values at node p_i. The data distance factor measures the similarity between the mean of the sensor readings at node p_i and the mean readings at its cluster head node p_j. It takes its maximum value of 1 when $V_i[p_j]$ is equal to μ_i. Its value decreases as the difference between $V_i[p_j]$ and μ_i increases, and approaches to 0 when the difference approaches to infinity. This is a generic way to calculate the data distance factor and does not require detailed knowledge about the data being collected. However, if such knowledge is available, a domain-specific data distance function can be applied. For instance, if a domain expert can set a system wide parameter Δ to be the maximum acceptable bound of the difference between the mean sample value of a node and the mean sample value of its head node, then we can specify a distance function $f(d) = d/\Delta$, where d is set to $|V_i[p_j] - \mu_i|$. In this case, the data distance factor can be calculated as $max(0, 1 - f(d))$. With this definition, the distance factor will take its maximum value of 1 when d is 0, and its value will linearly decrease to 0 as d reaches Δ.

We compute the attraction score as a weighted sum of the hop distance factor and the data distance factor, where the latter is multiplied by a factor called *data importance factor*, denoted by α. α takes a value in the range $[0, \infty)$. A value of 0 means only hop distance is used for the purpose of clustering. Larger values result in a clustering that is more dependent on the distances between the mean sample values of the nodes. The pseudo code for cluster engagement step is given by the PICKCLUSTERS procedure in Alg. 1.

Cluster-connection tree formation: Each node cluster in the network not only elects a cluster head node but also forms a cluster connection tree during the cluster construction. Such cluster connection trees are used to accomplish the communication of nodes with their cluster heads. Concretely, the cluster-connection trees are formed as follows: When a node p_i receives a cluster formation message originated at a cluster head node p_j, p_i notes down the node from which it has received this cluster formation message as the candidate for becoming the parent node of p_i in the cluster connection tree anchored at p_j. If there are several distinct cluster head nodes that circulate a cluster formation message to node p_i, then for each one of such cluster heads, say p_j, p_i stores only one forwarder node, say p_u, which is the one that has forwarded the cluster formation message of p_j with the highest TTL value. Now this forwarder node p_u becomes the parent pointer

of p_i for cluster C_j. After nodes have decided on which of the node clusters to join at the end of the cluster engagement step, each node sends a confirmation message to its corresponding cluster head using the parent pointer stored for that cluster. If a node p_i is in cluster C_j **or** has forwarded a confirmation message destined to the cluster head node p_j, then it keeps its parent pointer associated with C_j and at the same time it also records the nodes from which it has received a confirmation message destined to the cluster head node p_j as its child pointers in the cluster connection tree rooted at p_j. Any parent pointers that are not used to forward a confirmation message can be dropped to minimize the state kept for maintaining cluster-connection trees. All nodes that keep their parent pointers for cluster C_j are part of the cluster-connection tree of C_j. It is important to note that a node that participates in the cluster-connection tree of a node cluster C_j may **not** necessarily be a member of C_j, i.e. it is possible that $h_i \neq p_j$. This property ensures that with the cluster-connection tree of a node cluster C_j, its cluster head p_j can reach all nodes in C_j and vice versa.

3.3 Effect of α on Clustering

We use an example scenario to illustrate the effect of α on clustering. Figure 3 (a) shows an arrangement of 100 sensor nodes and the connectivity graph of the sensor network formed by these nodes. In the background, it also shows a colored image that represents the environmental values that are sensed by the nodes. The color on the image changes from tones of red (light gray in grayscale) to tones of blue (dark gray in grayscale) moving diagonally from the upper right corner to the lower left corner, representing a decrease in the sensed values. Figures 3 (b), (c), and (d) show three different clusterings of the network for different values of α (0, 10, 20 respectively), using $f_c = 0.1$ and $TTL = 5$. Each cluster is shown with a different color. Nodes within the same cluster are labeled with a unique cluster identifier. The cluster head nodes are marked with a circle. It is clearly observed that, with increased α, the clusters tend to align diagonally, resulting in a clustering where nodes sampling similar values are assigned to the same clusters. However, this effect is limited by the value of TTL, since a node cannot belong to a cluster whose cluster head is more that TTL hops away. We provide a quantitative study on the effect of α on the quality of the clustering in Section 7.

From both Algorithm 1 and Figure 3(c) and (d), one can observe that combining hop distance factor and sensor reading similarity captured by data distance factor, some of the resulting clusters may appear disconnected. As discussed in the previous section, by creating a cluster-connection tree for each node cluster, we guarantee that a cluster head node can reach all nodes in its cluster. When the number of connected subcomponents within a cluster is large, the overhead for a cluster head to communicate with nodes within its cluster will increase. However, the number of connected subcomponents of a sensing-driven cluster can not be large in practice due to three major reasons: First, since there is a fixed TTL value used in cluster creation, the nodes that belong to the same cluster can not be more than a specified number of hops away, thus it is

not possible that two nodes from different parts of the network are put within the same cluster just because the values they are sensing are very similar. Second, the decision to join a cluster is not only data dependent. Instead, it is a combination (adjusted by α) of hop distance factor and data distance factor that defines a node's affinity to join a cluster. As a result, unless TTL and α values are both set to impractically large values, there won't be many connected components belonging to the same cluster. Finally, since the sensor readings are expected to be spatially correlated, it is unlikely to have contiguous regions with highly heterogeneous sensor readings (which would have resulted in clusters with many connected subcomponents).

3.4 Setting of Clustering Period τ_c

The setting of clustering period τ_c involves two considerations. First, the cluster head nodes have additional responsibilities when compared to other nodes, due to sampler selection and model derivation process (see more detail in the next section), which causes them to consume energy at higher rates. Therefore, large τ_c values may result in imbalanced power levels and decrease network connectivity in the long run. Consequently, the value of τ_c parameter should be small enough to enable selection of alternate nodes as cluster heads. However, its value is expected to be much larger than the desired sampling and forced sampling periods, τ_d and τ_f. Second, time dependent changes in sensor readings may render the current clustering obsolete with respect to data distance factor. For instance, in environmental monitoring applications, different times of a day may result in different node clusters. Thus, clustering period should be adjusted accordingly to enable continued refinement of the clustering structure in response to different sensing patterns resulting from environmental changes.

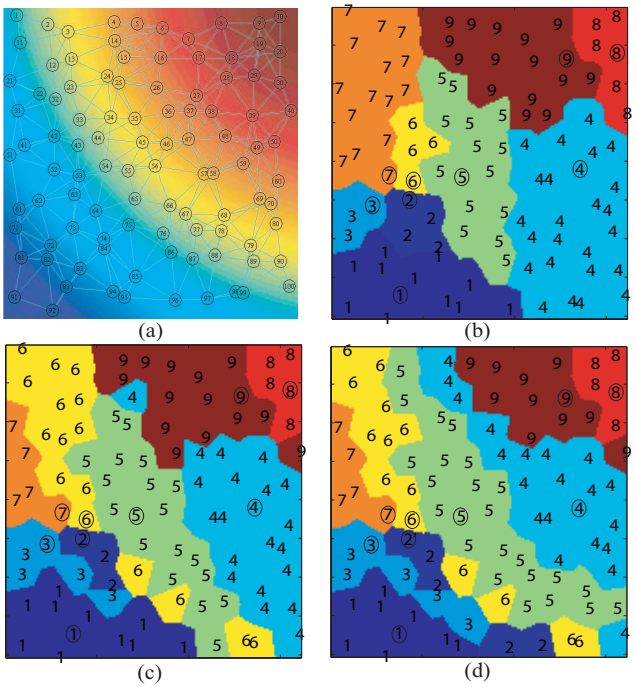

Figure 3: Illustration of sensing-driven clustering

4 Correlation-based Sampler Selection and Model Derivation

The goal of sampler selection and model derivation is three folds. First, it needs to further group nodes within each node cluster into a set of subclusters such that the sensor readings of the nodes within each subcluster are highly correlated (thus prediction is more effective). Second, it needs to derive and report (to sampler nodes) a sampling schedule that defines the sampler nodes. And third, it needs to derive and report (to the base node) parameters of the probabilistic models associated with each subcluster so that prediction can be

performed. Correlation-based sampler selection and model derivation is performed by each cluster head node through a three-step process, namely *subclustering*, *sampler selection*, and *model and schedule reporting*. We now describe these three steps in detail.

Subclustering step is used to create subclusters that form a basis for selecting sampler nodes of the network, deriving correlations among nodes within subclusters, constructing probabilistic models accordingly, and performing value predication of non-sampler nodes. Higher correlations among the nodes within each subcluster typically lead to higher quality sampler selection and higher accuracy of model-based value prediction of non-sampler nodes. Thus, given a cluster, the first issue involved in developing an effective subclustering algorithm is to obtain samples from nodes within this cluster. The second issue is to compute correlations between every pair of nodes within the cluster and define a correlation distance metric that can be used as the distance function for subclustering.

Notation	Meaning
D_i	Forced samples collected at node $p_i \in H$, where $D_i[p_j]$ is the series of consecutive forced samples from node $p_j \in C_i$
$\mathcal{C}_i$	Correlation matrix at node $p_i \in H$, where $\mathcal{C}_i[p_u, p_v]$ is the correlation between the series $D_i[p_u]$ and $D_i[p_v]$
$\mathcal{D}_i$	Subclustering distance matrix at node $p_i \in H$, where $\mathcal{D}_i[p_u, p_v]$ is the subclustering distance between p_u and p_v
β	Subcluster granularity
σ	Sampling fraction
τ_u	Schedule update period

Table 3: Notations for correlation-based sampler selection and model derivation notations

Forced sampling. Recall from Section 2.2, we introduce the concept of forced sampling period. By periodically collecting sample readings from all nodes in the network (forced sampling), the cluster head nodes can refine the subclustering structure by invoking a new run of the subclustering process, which utilizes the forced samples collected during the most recent clustering period to generate a new set of subclusters, each associated with a newly derived correlation-based probabilistic model. We denote the forced samples collected at cluster head node p_i by D_i. $D_i[p_j]$ denotes the series of consecutive forced samples from node p_j, where p_j is in the node cluster with p_i as the head (i.e. $p_j \in C_i$).

Correlation matrix and correlation distance metric. During subclustering, a cluster head node p_i takes the following concrete actions. It first creates a correlation matrix $\mathcal{C}_i$ such that for any two nodes in the cluster C_i, say p_u and p_v, $\mathcal{C}_i[p_u, p_v]$ is equal to the correlation between the series $D_i[p_u]$ and $D_i[p_v]$, formally $\frac{(D_i[p_u]-\mathrm{E}[D_i[p_u]])*(D_i[p_v]-\mathrm{E}[D_i[p_v]])^T}{L*\sqrt{\mathrm{Var}(D_i[p_u])}*\sqrt{\mathrm{Var}(D_i[p_v])}}$, where L is the length of the series and ^{T} represents matrix transpose. This is a textbook definition [10] of correlation between two series, expressed using the notations introduced within the context of this work. Correlation values are always in the range $[-1, 1]$, -1 and 1 representing strongest negative and positive correlation. A value of 0 implies two series are not correlated. As a result, the absolute correlation can be used as a metric to define how good two nodes are, in terms of predicting one's sample from another's. For each node cluster, we first compute its correlation matrix using forced samples. Then we calculate the correlation distance metric between nodes, denoted by $\mathcal{D}_i$. $\mathcal{D}_i[p_u, p_v]$ is defined as $1 - |\mathcal{C}_i[p_u, p_v]|$. Once we get the distance metric, we use agglomerative clustering [21] to subcluster the nodes within clus-

ter C_i into K_i number of subclusters, where $G_i(j)$ denotes the set of nodes in the jth subcluster. We use a system-wide parameter called *subcluster granularity*, denoted by β, to define average subcluster size. Thus, K_i is calculated by $\lceil |C_i|/\beta \rceil$. We'll discuss the effects of β on performance later in this section. The pseudo code for the subclustering step is given within the SUBCLUSTERANDDERIVE procedure in Alg. 2.

Sampler selection step is performed to create or update a data collection schedule S_i for each cluster C_i, in order to select the subset of nodes that are best qualified to serve as samplers throughout the next schedule update period τ_u. After a cluster head node p_i forms the subclusters, it initializes the data collection schedule S_i to zero for all nodes within its cluster, i.e. $S_i[p_j] = 0, \forall p_j \in C_i$. Then for each subcluster $G_i(j)$, it determines the number of sampler nodes to be selected from that subcluster based on the size of the subcluster $G_i(j)$ and the sampling fraction parameter σ defined in Section 2. At least one node should be selected as a sampler node from each subcluster. Thus we can calculate the number of sampler nodes for a given subcluster $G_i(j)$ by $\lceil \sigma * |G_i(j)| \rceil$. Based on the above formula, we can calculate the actual fraction of nodes selected as the sampler nodes of the network at any given instance of time. This actual fraction may deviate from the system-supplied sampling fraction parameter σ. We refer to the actual fraction of sampler nodes as the *effective* σ to distinguish it from the system-supplied σ. The *effective* σ can be estimated as $f_c * \lceil 1/(f_c * \beta) \rceil * \lceil \beta * \sigma \rceil$. The pseudo code for the derivation step is given within the SUBCLUSTERANDDERIVE procedure in Alg. 2.

Alg. 2: Corr.-based Sampler Select. and Model Derivation

```
DERIVESCHEDULE(p_i ∈ H)
(1)  Periodically, every τ_u seconds
(2)      D_i: data collected since last schedule derivation, D_i[p_j](k) is the
         kth forced sample from node p_j collected at node p_i
(3)      (S_i, C_i, G_i) ← SUBCLUSTERANDDERIVE(p_i, D_i)
(4)      for j = 1 to |G_i|
(5)          X_{i,j} : X_{i,j}[p_u] = E[D_i[p_u]]; p_u ∈ G_i(j)
(6)          Y_{i,j} : Y_{i,j}[p_u, p_v] = C_i[p_u, p_v] * sqrt(Var(D_i[p_u])) *
             sqrt(Var(D_i[p_v])); p_u, p_v ∈ G_i(j)
(7)          SENDMSG(base, X_{i,j}, Y_{i,j})
(8)      foreach p_j ∈ C_i
(9)          D_i[p_j] ← ∅
(10)         SENDMSG(p_j, S_i[p_j])

SUBCLUSTERANDDERIVE(p_i ∈ H)
(1)  ∀p_u, p_v ∈ C_i, C_i[p_u, p_v] ← Correlation between D_i[p_u], D_i[p_v]
(2)  ∀p_u, p_v ∈ C_i, D_i[p_u, p_v] ← 1 − |C_i[p_u, p_v]|
(3)  K_i ← ⌈|C_i|/β⌉    /* number of subclusters */
(4)  Cluster the nodes in C_i, using D_i as distance metric, into K_i
     subclusters
(5)  G_i(j) : nodes in the jth subcluster within C_i, j ∈ {1, ..., K_i}
(6)  t ← Current time
(7)  ∀p_u ∈ C_i, S_i[p_u] ← 0
(8)  foreach j ∈ {1, ..., K_i}
(9)      a ← ⌈σ * |G_i(j)|⌉
(10)     foreach p_u ∈ G_i(j), in decreasing order of e_u(t)
(11)         S_i[p_u] ← 1
(12)         if a = |{p_v| S_i[p_u] = 1}| then break
(13) return (S_i, C_i, G_i)
```

Model and schedule reporting step is performed by a cluster head node in two steps, after generating the data collection schedule for each node cluster. First, the cluster head informs the nodes about their status as samplers or non-samplers. Then the cluster head sends the summary information to the base node, which will be used to derive the parameters of probabilistic models used in predicting the values of non-sampler nodes. To implement the first step, a cluster head node p_i notifies each node p_j within its cluster about p_j's new status with regard to being a sampler node or not by sending $S_i[p_j]$ to p_j. To realize the second step, for each subcluster $G_i(j)$, p_i calculates a data mean vector for nodes within the subcluster, denoted by $\mathcal{X}_{i,j}$, as follows:

$\mathcal{X}_{i,j}[p_u] = \mathrm{E}[D_i[p_u]], p_u \in G_i(j)$. p_i also calculates a data covariance matrix for nodes within the subcluster, denoted by $\mathcal{Y}_{i,j}$ and defined as follows: $\mathcal{Y}_{i,j}[p_u, p_v] = \mathcal{C}_i[p_u, p_v] * \sqrt{\mathrm{Var}(D_i[p_u])} * \sqrt{\mathrm{Var}(D_i[p_v])}, p_u, p_v \in G_i(j)$. For each subcluster $G_i(j)$, p_i sends $\mathcal{X}_{i,j}$, $\mathcal{Y}_{i,j}$ and the identifiers of the nodes within the subcluster, to the base node. This information will later be used for deriving the parameters of a Multi-Variate Normal (MVN) model for each subcluster (see Section 5). The pseudo code is given within the DERIVESCHEDULE procedure of Alg. 2.

4.1 Effects of β on Performance

The setting of the system supplied parameter β (subcluster granularity) may have effects on the overall performance of a selective sampling based data collection system, especially in terms of sampler selection quality, value predication quality, messaging cost, and energy consumption. Intuitively, large values of β may decrease the prediction quality, because it will result in large subclusters with potentially low overall correlation between its members. On the other hand, too small values may also decrease the prediction quality, since the opportunity to exploit the spatial correlations fully will be missed with very small β. Regarding the messaging cost of sending sampling summarization and model derivation information to the base node, one extreme case is where each cluster has one subcluster (very large β). In this case, the covariance matrix may become very large and sending it to the base station may increase the messaging cost and have a negative effect on the energy-efficiency. In contrast, smaller β values will result in a lower messaging cost, since covariance values of node pairs belonging to different subclusters will not be reported. Although the the second dimension favors a small β value, decreasing beta will increase the deviation of effective σ from the system specified σ, introducing another dimension. For instance, having $\beta = 2$ will result in a minimum effective σ of around 0.5, even if σ is specified much smaller. This is because each subcluster must have at least one sampler node. Consequently, the energy saving expected when σ is set to a certain value is dependent on the setting of β. In summary, small β values can make it impossible to practice high energy saving/low prediction quality scenarios. We investigate these issues quantitatively in Section 7.

4.2 Setting of Schedule Update Period τ_u

The schedule update period τ_u is a system supplied parameter and it defines the time interval for re-computing the subclusters of a node cluster in the network. Several factors may affect the setting of τ_u. First, the nodes that are samplers consume more energy compared to non-samplers, since they perform sensing and report their sensed values. Consequently, the value of τ_u parameter should be small enough to enable selection of alternate nodes as samplers through the use of energy-aware schedule derivation process, in order to balance power levels of the nodes. Moreover, such alternate node selections help in evenly distributing the error of

prediction among all the nodes. As a result, τ_u is provisioned to be smaller compared to τ_c, so that we can provide fine-grained sampler re-selection without much overhead. Second, the correlations among sensor readings of different nodes may change with time and deteriorate the prediction quality. As a result, the schedule update period should be adjusted accordingly, based on dynamics of the specific application at hand.

5 Selective Data Collection and Model-based Prediction

Our selective sampling approach achieves energy efficiency of data collection services by collecting sample readings from only a subset of nodes (sampler nodes) that are carefully selected and dynamically changing (after every schedule update period). The values of non-sampler nodes are predicted using probabilistic models whose parameters are derived from the recent samples of nodes that are spatially and temporally correlated. The energy saving is a result of smaller number of messages used to extract and collect data from the network, which is a direct benefit of smaller number of sensing operations performed. Although all nodes have to sample after every forced sampling period (recall that these samples are used for predicting the parameters of MVN models for the subclusters), these forced samples do not propagate up to the base node, and are collected locally at cluster head nodes. Instead, only a summary of the model parameters are submitted to the base node after each correlation-based model derivation step.

Notation	Meaning
$\mathcal{X}_{i,j}$	Data mean vector for nodes in $G_i(j)$, where $\mathcal{X}_{i,j}[p_u]$ is the mean of the forced samples from node $p_u \in G_i(j)$.
$\mathcal{Y}_{i,j}$	Data covariance matrix for nodes in $G_i(j)$, where $\mathcal{Y}_{i,j}[p_u, p_v]$ is the covariance between the series $D_i[p_u]$ and $D_i[p_v]$
$U^+_{i,j}$	Set of nodes belonging to $G_i(j)$ that are samplers
$U^-_{i,j}$	Set of nodes belonging to $G_i(j)$ that are not samplers
$W^+_{i,j}$	Set of last reported sensor readings of nodes in $U^+_{i,j}$
$W^-_{i,j}$	Set of predicted sensor readings of nodes in $U^-_{i,j}$
τ_d	Desired sampling period
τ_f	Forced sampling period

Table 4: Notations for Selective data collection and model-based prediction parameters

In effect, one sample value from every node is calculated at the base node (or at the sensor stream processing center). However, a sample value comes from either a *direct* sample or a *predicted* sample. Direct samples are the ones that originate from actual sensor readings. If a node p_i is a sampler, i.e. $S_j[p_i] = 1$ where $h_i = p_j$, it periodically reports its sensor reading to the base node using the data collection tree, i.e. after every desired sampling period τ_d, except when forced sampling and desired sampling periods coincide (recall that τ_f is a multiple of τ_d). In the latter case, the sensor reading is sent to the cluster head node h_i using the cluster-connection tree, and is forwarded to base node from there. If a node p_i is a non-sampler node, i.e. $S_j[p_i] = 0$ where $h_i = p_j$, then it only samples after every forced sampling period, and its sensor readings are sent to the cluster head node h_i using the cluster-connection tree and are *not* forwarded to the base node. A short pseudo code describing this is given by the SENSDATA procedure in Alg. 3.

5.1 Calculating predicted sample values

The problem of predicting the sample values of non-sampler nodes can be described as follows. Given a set of sample values belonging to same sampling step from sampler nodes within a subcluster $G_i(j)$, how can we predict the set of sample values belonging to non-sampler nodes within $G_i(j)$, given the mean vector $\mathcal{X}_{i,j}$ and covariance matrix $\mathcal{Y}_{i,j}$ for the subcluster. We denote the set of sampler nodes from subcluster $G_i(j)$ by $U^+_{i,j}$, defined as $\{p_u|\ p_u \in G_i(j), S_i[p_u] = 1\}$. Similarly, we denote the set of non-sampler nodes by $U^-_{i,j}$, defined as $\{p_u|\ p_u \in G_i(j), S_i[p_u] = 0\}$. Let $W^+_{i,j}$ be the set of sample values from the same sampling step, received from the sampler nodes $U^+_{i,j}$. Using a MVN model to capture the spatial and temporal correlations within a subcluster, we utilize the following theorem that can be found in texts on statistical inference [10], to predict the values of the non-sampler nodes:

Theorem 1: *Let X be a MVN distributed random variable with mean μ and covariance matrix Σ. Let μ be partitioned as $\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$ and Σ partitioned as $\begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$. According to this, X is also partitioned as X_1 and X_2. Then the distribution of X_1 given $X_2 = A$ is also MVN with mean $\mu^* = \mu_1 + \Sigma_{12} * \Sigma_{22}^{-1} * (A - \mu_2)$ and covariance matrix $\Sigma^* = \Sigma_{11} - \Sigma_{12} * \Sigma_{22}^{-1} * \Sigma_{21}$.*

In accordance with the theorem, we construct μ_1 and μ_2 such that they contain the mean values in $\mathcal{X}_{i,j}$ that belong to nodes in $U^-_{i,j}$ and $U^+_{i,j}$, respectively. A similar procedure is performed to construct $\Sigma_{11}, \Sigma_{12}, \Sigma_{21}$, and Σ_{22} from $\mathcal{Y}_{i,j}$. Σ_{11} contains a subset of $\mathcal{X}_{i,j}$ which describes the covariance among the nodes in $U^-_{i,j}$, and Σ_{22} among the nodes in $U^+_{i,j}$. Σ_{12} contains a subset of $\mathcal{X}_{i,j}$ which describes the covariance between the nodes in $U^-_{i,j}$ and $U^+_{i,j}$, and Σ_{21} is its transpose. Then the theorem can be directly applied to predict the values of non-sampler nodes $U^-_{i,j}$, denoted by $W^-_{i,j}$. $W^-_{i,j}$ can be set to $\mu^* = \mu_1 + \Sigma_{12} * \Sigma_{22}^{-1} * (W^+_{i,j} - \mu_2)$, which is the maximum likelihood estimate, or $\mathcal{N}(\mu^*, \Sigma^*)$ can be used to predict the values with desired confidence intervals. We use the former in the rest of the paper. The details of prediction step is given by the PREDICTDATA procedure in Alg. 3.

Alg. 3: Selective Data Collection and Model-based Prediction

```
SENSDATA(p_i)
(1)  if S_j[p_i] = 0, where h_i = p_j
(2)      Periodically, every τ_f seconds
(3)          d_i ← SENSE()
(4)          SENDMSG(h_i, d_i)
(5)  else
(6)      Periodically, every τ_d seconds
(7)          d_i ← SENSE()
(8)          t ← Current time
(9)          if mod(t, τ_f) = 0
(10)             SENDMSG(h_i, d_i)
(11)         else
(12)             SENDMSG(base, d_i)

PREDICTDATA(i, j, U^+, U^-, W^+)
U^+ = {p_{u_1^+}, ..., p_{u_k^+}} : set of nodes from jth subcluster in C_i whose data values are received
U^- = {p_{u_1^-}, ..., p_{u_l^-}} : set of nodes from jth subcluster in C_i whose data values are missing
W^+ : W^+(a), a ∈ {1, ..., k} is the value reported by node p_{u_a^+}
(1)  X_{i,j}: mean vector for jth subcluster in C_i
(2)  Y_{i,j}: covariance matrix for jth subcluster in C_i
(3)  for a = 1 to l
(4)      μ_1(a) ← X_{i,j}[p_{u_a^-}]
(5)      for b = 1 to l, Σ_11(a, b) ← Y_{i,j}[p_{u_a^-}, p_{u_b^-}]
(6)      for b = 1 to k, Σ_12(a, b) ← Y_{i,j}[p_{u_a^-}, p_{u_b^+}]
(7)  for a = 1 to k
(8)      μ_2(a) ← X_{i,j}[p_{u_a^+}]
(9)      for b = 1 to k, Σ_22(a, b) ← Y_{i,j}[p_{u_a^+}, p_{u_b^+}]
(10) μ* = μ_1 + Σ_12 * Σ_22^{-1} * (W^+ − μ_2)
(11) Σ* = Σ_11 − Σ_12 * Σ_22^{-1} * Σ_12^T
(12) Use N(μ*, Σ*) to predict values of nodes in U^-
```

5.2 Prediction Models

The detailed algorithm governing the prediction step can consider alternative inference methods and/or statistical models with their associated parameter specifications, in addition to the prediction method described in this section and the Multi-Variate Normal model used with data mean vector and data covariance matrix as its parameters. Our data collection framework is flexible enough to accommodate such alternative prediction methodologies. For instance, we can keep the MVN model and change the inference method to Bayesian inference. This can provide significant improvement in prediction quality if prior distributions of the samples are available or can be constructed from historical data. This flexibility allows us to understand how different statistical inference methods may impact the quality of the model-based prediction. We can go one step further and change the statistical model used, as long as the model parameters can be easily derived locally at the cluster heads and are reasonably compact in size.

5.3 Setting of Forced and Desired Sampling Periods τ_f and τ_d

The setting of forced sampling period τ_f involves three considerations. First, increased number of forced samples (thus smaller τ_f) may be desirable, since it can improve the ability to capture correlations in sensor readings better. Second, large number of forced samples can cause the memory constraint on sensor nodes to be a limiting factor, since the cluster head nodes are used to collect forced samples. Pertaining to this, a lower bound on τ_f can be computed based on the number of nodes in a cluster and the schedule update period τ_u. For instance, if we want the forced samples to occupy an average memory size of M units where each sensor reading occupy R units, then we should set τ_f to a value larger than $\frac{\tau_u * R}{f_c * M}$. Third, less frequent forced sampling results in smaller set of forced samples, which is more favorable in terms of messaging cost and overall energy consumption. In summary, the value of τ_f should be set taking into account the memory constraint and the desired trade-off between prediction quality and network lifetime. The setting of desired sampling period τ_d defines the temporal resolution of the collected data and is application specific.

6 Analytical Analysis of Overall Messaging Cost

In this section, we provide an analytical study on the messaging cost of data collection based on our selective sampling architecture. For the purpose of comparison, we introduce two variations of selective sampling – *central* approach and *local* approach. The central approach presents one extreme of the spectrum, in which both the model prediction and the value prediction of non-sampling nodes are carried out at the base node or processing center outside the network. This means that all forced samples are forwarded to the base node to compute the correlations in a centralized location. In the local approach, value prediction is performed at the

cluster heads instead of the base nodes or the processing center outside the network, and predicted values are reported to the base node. Although the local approach results in a large messaging cost and works against the idea of selective sampling, it can be used to serve as a base case for comparison. The selective sampling solution falls in between these two extremes. We thus call it the *hybrid* approach due to the fact that the spatial and temporal correlations are captured and summarized locally within the network, whereas the value prediction is performed centrally at the base node or outside the network.

In the rest of this section, we calculate the total number of messages sent and received (spent) within the network during a time interval of T seconds, denoted by M_t^h, M_t^c, and M_t^l, respectively for hybrid, central, and local approaches. We denote the total number of clusterings and subclusterings performed during the time interval of length T as N_{nc} and N_{ns}. We have $N_{nc} = \lfloor T/\tau_c \rfloor$ and $N_{ns} = \lfloor T/\tau_u \rfloor$. Similarly, the total number of forced samplings and desired samplings are denoted by N_{fs} and N_{ds}. We have $N_{fs} = \lfloor T/\tau_f \rfloor$ and $N_{ds} = \lfloor T/\tau_d \rfloor$.

The total number of messages can be broken into three components, namely messages spent during *i*) cluster construction, *ii*) schedule and model derivation, and *iii*) selective data collection. The messages spent during cluster construction, denoted by M_{tc}, is same for all approaches and can be defined as $M_{tc} = N_{nc} * M_{cs}$, where M_{cs} denotes the number of messages spent during one clustering step. Since N_{nc} is expected to be much smaller than N_{ns}, N_{fs}, and N_{ds}, we omit the derivation of M_{cs} in the interest of space[1]. Now we describe the derivation of the remaining two components *ii*) and *iii*), for three different scenarios. We use the notations I_i^b and I_i^c to denote the distance of node p_i (in terms of hops) to the base node and its cluster head node h_i, respectively. Each message is assumed to have the size of a basic message, where a basic message includes a node identifier and a sensor reading.

The derivation for the **Hybrid Approach** is as follows:

$$\begin{aligned} M_{ts}^h &= N_{ns} * \sum_{i=1}^{N} I_i^c + N_{ns} * \sum_{p_i \in H} \left(I_i^b * \sum_{j=1}^{K_i} \left(|G_i(j)| * \frac{3 + |G_i(j)|}{4} \right) \right) \\ M_{tm}^h &= N_{fs} * \sum_{p_i \in H} \sum_{p_j \in C_i} (I_j^c + S_i[p_j] * I_i^b) + (N_{ds} - N_{fs}) * \sum_{p_i \in H} \sum_{p_j \in C_i} (S_i[p_j] * I_j^b) \\ M_t^h &= M_{tc} + M_{ts}^h + M_{tm}^h \end{aligned} \quad (1)$$

Here M_{ts}^h denotes the schedule and model derivation component for the hybrid approach. It consists of two subcomponents, messages spent for notifying each node after schedules are derived, and the messages spent for reporting the covariance matrix, mean vector, and the node identifiers to the base node for each subcluster. Note that the covariance matrix is symmetric and thus not all the entries are reported. M_{tm}^h denotes the data collection component for the hybrid approach. In summary, it counts the messages from sampler nodes

[1]However M_{tc} is accounted for in the experimental results of Section 7.

and non-sampler nodes. Messages from non-sampler nodes are forwarded up to the cluster heads after every forced sampling period. On the other hand, messages from sampler nodes are forwarded up to the base node after every desired sampling period (except when τ_d and τ_f coincide). Finally, the total number of messages for the hybrid approach, denoted by M_t^h, is calculated in Equation 1 as the sum of three components.

The derivation for the **Central Approach** is as follows:

$$\begin{aligned} M_{ts}^c &= N_{ns} * \sum_{i=1}^{N} I_i^c \\ M_{tm}^c &= N_{fs} * \sum_{i=1}^{N} I_i^b + (N_{ds} - N_{fs}) * \sum_{p_i \in H} \sum_{p_j \in C_i} (S_i[p_j] * I_j^b) \end{aligned} \qquad M_t^c = M_{tc} + M_{ts}^c + M_{tm}^c \tag{2}$$

Here M_{ts}^c denotes the schedule and model derivation component for the central approach. It consists of the messages used for notifying each node after schedules are derived. As opposed to hybrid scenario, it does *not* include the reporting of covariance matrices or mean vectors. M_{tm}^c denotes the data collection component for the central approach. Different from the hybrid scenario, all forced samples are forwarded up to the base node. Finally, the total number of messages for the central approach, denoted by M_t^c, is calculated in Equation 2 as the sum of three components.

The derivation for the **Local Approach** is as follows:

$$\begin{aligned} M_{ts}^l &= N_{ns} * \sum_{i=1}^{N} I_i^c \\ M_{tm}^l &= M_{tm}^h + N_{ds} * \sum_{p_i \in H} \sum_{p_j \in C_i} (I_i^b * (1 - S_i[p_j])) \end{aligned} \qquad M_t^l = M_{tc} + M_{ts}^l + M_{tm}^l \tag{3}$$

Here M_{ts}^l denotes the schedule and model derivation component for the local approach and is identical to the same component of the central approach. M_{tm}^l denotes the data collection component for the local approach. It can be considered as the data collection component of the hybrid approach, plus the number of messages spent for forwarding the predicted samples from the cluster head nodes to the base node. Finally, the total number of messages for the local approach, denoted by M_t^l, is calculated in Equation 3.

7 Performance Study

We present analytical and simulation based experimental results to study the effectiveness of our selective sampling approach. We divided the experiments into two sets. The first set of experiments compares different variations of selective sampling and studies the impact of various parameters on performance, with regard to messaging cost. These results are based on analytical derivations. The second set of experiments study the effect of various system parameters on the quality of collected data as well as the quality/lifetime trade-off. These experiments are based on simulations using real-world data.

7.1 Messaging Cost

We calculate the total number of messages spent for data collection using different approaches, namely hybrid, central, local and non-selective, and compare the results for different values of system parameters. The non-selective case refers to *naïve* periodic data collection with no support for selective sampling.

In general, the gap between local and non-selective approaches, with the local approach being more expensive in terms of messaging cost, indicates the overhead of cluster construction, sampler selection and model derivation, and selective data collection steps when the savings due to selective sampling are removed (local approach). On the other hand, the gap between central and hybrid approaches, with the hybrid being less expensive thus better, indicates the savings obtained by only reporting the summary of correlations among sensor nodes within each of the subclusters (hybrid approach), instead of forwarding all forced samples to the base node (central approach). The default parameters used in this set of experiments are as follows. The total time is set to $T = 1000000$ units. The total number of nodes in the network is set to 600 unless specified otherwise. f_c is selected to result in an average cluster size of 30 nodes. Desired and forced sampling periods are set to $\tau_d = 1$ and $\tau_f = 10$ time units. Clustering period is set to $\tau_c = 5000$ time units and the schedule update period is set to $\tau_u = 1000$ time units. Sampling fraction σ is set to 0.25 and β is set to 10.

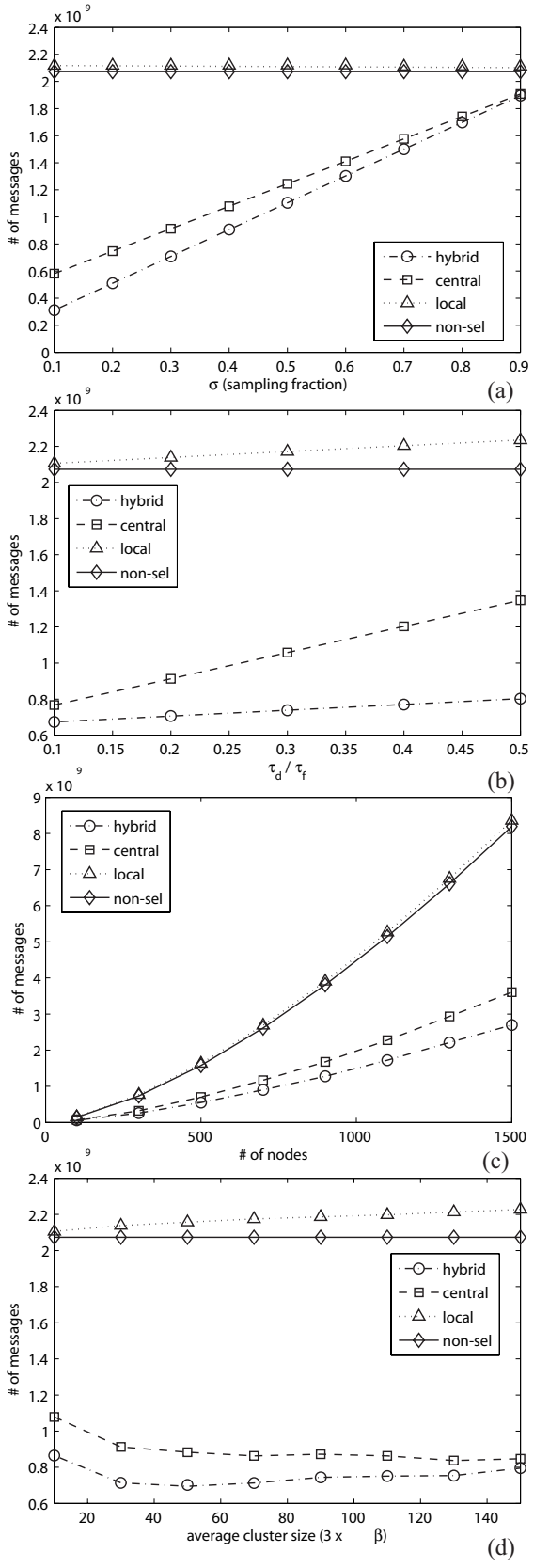

Figure 4: Analysis of messaging cost

Figure 4(a) plots the total number of messages as a function of the sampling fraction σ. We make several observations from the figure. First, as expected, central and hybrid approaches provide significant improvement over local and non-selective approaches. This improvement decreases as σ increases, since increasing values of σ imply that larger number of nodes are becoming samplers. Second, the overhead of schedule and model derivation step can be observed by comparing

non-selective and local approaches. Note that the gap between the two is very small and implies that this step incurs very small messaging overhead. Third, the improvement provided by the hybrid approach can be observed by comparing hybrid and central approaches. We see an improvement ranging from 50% to 12% to around 0%, while σ increases from 0.1 to 0.5 to 0.9. This shows that the hybrid approach is superior to the central approach and is effective in terms of messaging cost especially when σ is small.

Figure 4 (b) plots the total number of messages as a function of the desired sampling period to forced sampling period ratio (τ_d/τ_f). In this experiment τ_d is fixed at 1 and τ_f is altered. We make two observations. First, there is an increasing overhead in the total number of messages with increasing τ_d/τ_f, as it is observed from the gap between local and non-selective approaches. This is mainly due to the increasing number of forced samples, which results in higher number of values from sampler nodes to first visit the cluster head node and then reach the base node, causing an overhead compared to forwarding values directly to the base node. Second, we observe that the hybrid approach prevails over other alternatives and provides an improvement over central approach, ranging from 10% to 42% while τ_d/τ_f ranges from 0.1 to 0.5. This is because the forced samples are only propagated up to the cluster head node with the hybrid approach.

Figure 4 (c) plots the total number of messages as a function of the number of nodes. The main observation from the figure is that, central and hybrid approaches scale better with increasing number of nodes, where the hybrid approach keeps its relative advantage over central approach (around %25 in this case) for different network sizes. Figure 4 (d) plots the total number of messages as a function of the average cluster size (i.e. $1/f_c$). β is also increased as the average cluster size is increased, so that the average number of subclusters per cluster is kept constant (around 3). From the gap between local and non-selective approaches, we can see a clear overhead that increases with cluster size. On the other hand, this increase does not cause an overall increase in the messaging cost of the hybrid approach until the average cluster size increases well over its default value of 30. It is observed from the figure that the best value for the average cluster size is 50 for this scenario, where smaller and larger values increase the messaging cost. It is also interesting to note that in the extreme case, where there is a single cluster in the network, central and hybrid approaches should converge. This can be observed from the right end of the x-axis in the figure.

7.2 Data Collection Quality

We study the data collection quality of our selective sampling approach through a set of simulation based experiments using real data. In particular, we study the effect of α on the quality of clustering, the effect of α, β and subclustering methodology on the prediction error, the trade-off between network lifetime (energy saving) and prediction error, and the load balance in selective sampling schedule derivation. For the purpose of experiments presented in this section, 1000 sensor nodes are placed in a square grid with a side length of

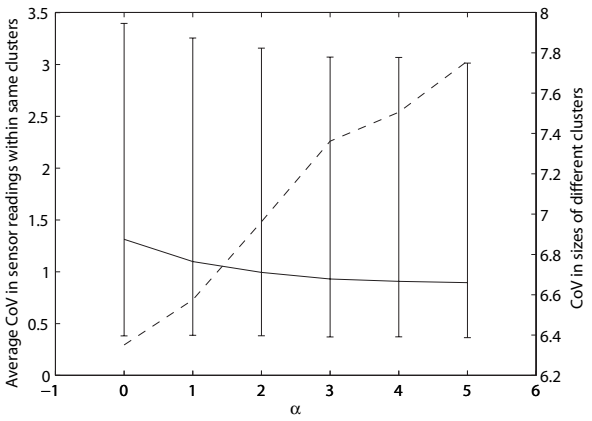

Figure 5: Clustering quality with varying α

α	Mean Absolute Deviation (Relative)	%90 Confidence Interval
0	0.3909 (0.1840)	[0.0325, 2.5260]
1	0.3732 (0.1757)	[0.0301, 2.0284]
2	0.3688 (0.1736)	[0.0296, 1.9040]
3	0.3644 (0.1715)	[0.0290, 1.7796]
4	0.3600 (0.1695)	[0.0284, 1.6552]

Table 5: Error for different α values

1 unit and the connectivity graph of the sensor network is constructed assuming that two nodes that are at most 0.075 units away from each other are neighbors. Settings of other relevant system parameters are as follows. TTL is set to 5. The sampling fraction σ is set to 0.5. β is set to 5 and f_c is set to 0.02 resulting in an average cluster size of 50. The data set used for the simulations is derived from the GPCP One-Degree Daily Precipitation Data Set (1DD Data Set) [19]. It provides daily, global 1x1-degree gridded fields of precipitation measurements for the 3-year period starting from January 1997. This data is mapped to our unit square and a sensor reading of a node at time step i is derived as the average of the five readings from the ith day of the 1DD data set whose grid locations are closest to the location of the sensor node (since the dataset has high spatial resolution).

Effect of α on the Quality of Clustering: Figure 5 plots the average coefficient of variance (CoV) of sensor readings within same clusters (with a solid line using the left y-axis), for different α values. For each clustering, we calculate the mean, maximum and minimum of the CoV values of the clusters, where CoV of a cluster is calculated over the mean data values of sensor nodes within the cluster. Averages from several clusterings are plotted as an error bar graph in Figure 5, where the two ends of the error bars correspond to average minimum and average maximum CoV values. Smaller CoV values in sensor readings imply a better clustering, since our aim is to gather together sensor nodes whose readings are similar. We observe that increasing α from 0 to 4 decreases the CoV around %50, where further increase in α do not provide improvement for this experimental setup. To show the interplay between the shape of the clusters and sensing-driven clustering, Figure 5 also plots the CoV in the sizes of clusters (with a dashed line using the right y-axis). With hop based clustering (i.e., $\alpha = 0$), the cluster sizes are expected to be more evenly distributed when compared to sensing-driven clustering. Consequently, the CoV in the sizes of clusters increases with increasing α, implying that the shape of clusters are being influenced by the similarity of sensor readings. These results are in line with our visual inspection based results shown in Figure 3 in Section 3.3.

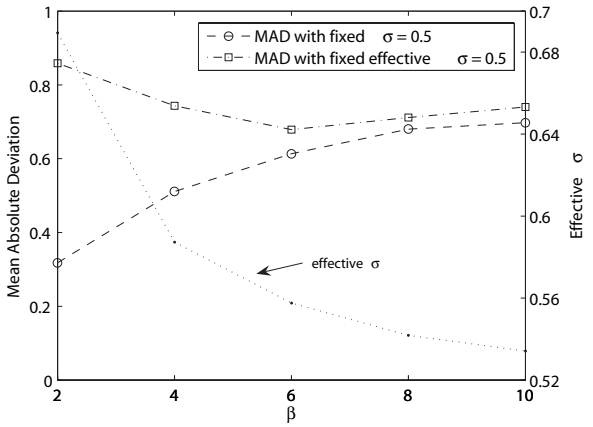

Figure 6: Effect of β on prediction error

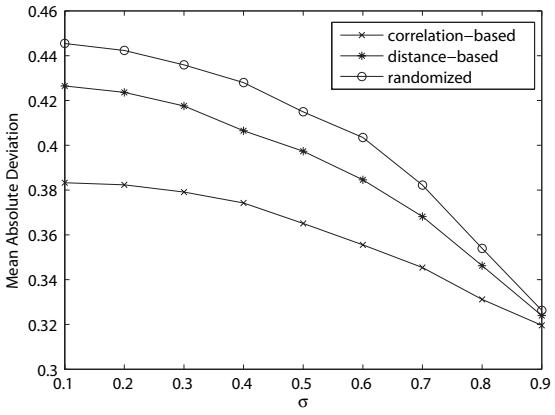

Figure 7: MAD with different subclusterings

Effect of α on the Prediction Error: In order to observe the impact of data-centric clustering on prediction quality, we study the effect of increasing data importance factor α on the prediction error. The second column of Table 5 lists the mean absolute deviation (MAD) of the error in predicted sample values for different α values listed in the first column. The value of MAD relative to the mean of the data values (2.1240) is also given within parenthesis in the first column. Although we observe a small improvement around 1% in the relative MAD when α is increased from 0 to 4, the improvement is much more prominent when we examine the higher end of the 90% confidence interval of absolute deviation, given in the third column of Table 5. The improvement is around 0.87, which corresponds to an improvement of 25% relative to the data mean.

Effect of β on the Prediction Error: As mentioned in Section 4.1, decreasing subcluster granularity parameter β is expected to increase effective σ. Higher effective σ implies larger number of sampler nodes and thus improves the error in prediction. Figure 6 illustrates this inference concretely, where the mean absolute derivation (MAD) of the error in predicted sample values and effective σ are plotted as a function of β. MAD is plotted with a dashed line and is read from the left y-axis, whereas effective σ is plotted with a dotted line and is read from the right y-axis. We see that decreasing β from 10 to 2 decreases MAD around 50% (from 0.44 to 0.22). However, this is mainly due to the fact that the average number of sampler nodes is increased by 26% (0.54 to 0.68). To understand the impact of β better and to decouple it from the number of sampler nodes, we fix effective σ to 0.5. Figure 6 plots MAD as a function of β for fixed effective σ, using a dash-dot line. It is observed that both small and large β values result in higher MAD whereas moderate values for β achieve smaller MAD. This is very intuitive, since small sized models (small β) are unable to fully exploit the available correlations between node samples, whereas large sized models (large β) become ineffective due to decreased amount of correlation among the samples of large and diverse node groups.

Effect of Subclustering on the Prediction Error: This experiment intends to show how different subclustering methods can affect the prediction error. We consider three different methods: correlation-based sub-

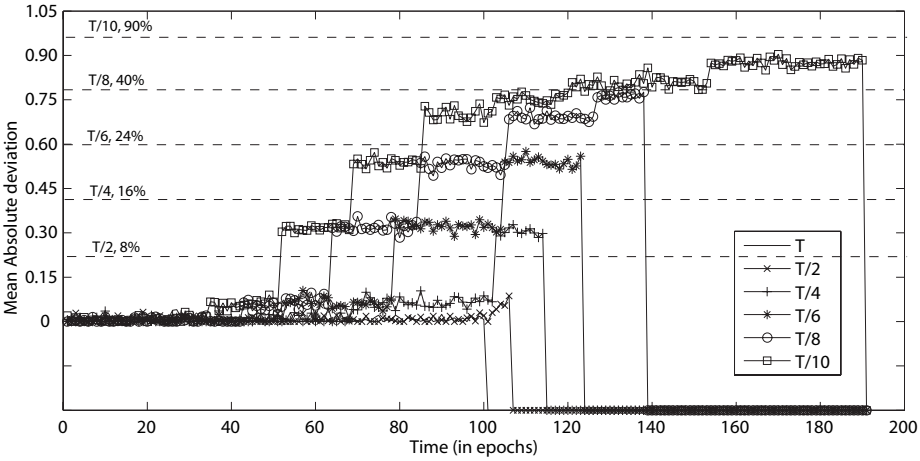

Figure 8: Prediction error vs. lifetime trade-off

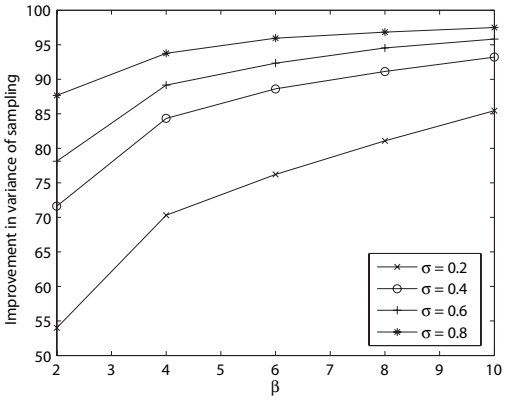

Figure 9: Load balance in schedule derivation

clustering (as described in Section 4), distance-based subclustering in which location closeness is used as the metric for deciding on subclusters, and randomized subclustering which uses purely random assignment to form subclusters. Figure 7 plots MAD as a function of σ for these three different methods of subclustering. The results listed in Figure 7 are averages of large number of subclusterings. We observe that randomized and distance-based subclustering perform up to 15% and 10% worse respectively, when compared correlation-based subclustering, in terms of mean absolute deviation of the error in value predication. The differences between these three methods in terms of MAD is largest when σ is smallest and disappears as σ approaches 1. This is quite intuitive, since smaller σ values imply that the prediction is performed with smaller number of sampler node values.

Prediction Error/Lifetime Trade-off: We study the trade-off between prediction error and network lifetime by simulating selective sampling with dynamic σ adjustment for different σ reduction rates. We assume that the main source of energy consumption in the network is wireless messaging and sensing. We set up a scenario such that, without selective sampling the average lifetime of the network is $T = 100$ units. This means that, the network enables us to collect data with 100% accuracy for 100 time units and then dies out. For comparison, we use selective sampling and experiment with dynamically decreasing σ as time progresses, in order to gradually decrease the average energy consumption, while introducing an increasing amount of error in the collected data. Figure 8 plots the mean absolute deviation (MAD) as a function of time for different σ reduction rates. In the figure $T/x, x \in \{1, 2, 4, 6, 8, 10\}$ denotes different reduction rates, where σ is decreased by 0.1 every T/x time units. σ is not dropped below 0.1. A negative MAD value in the figure implies that the network has exceeded its lifetime. Although it is obvious that the longest lifetime is achieved with the highest reduction rate (easily read from the figure), most of the time it is more meaningful to think of lifetime as bounded by the prediction error. In other words, we define the ϵ-*bounded network lifetime* as the longest period during which the MAD is always below a user defined threshold ϵ. Different thresholds are

plotted as horizontal dashed lines in the figure, crossing the y-axis. In order to find the σ reduction rate with the highest ϵ-bounded network lifetime, we have to find the error line that has the largest x-axis coordinate (lifetime) such that its corresponding y-axis coordinate (MAD) is below ϵ and above zero. Following this, the approach with the highest ϵ-bounded lifetime is indicated over each ϵ line together with the improvement in lifetime. We observe that higher reduction rates do not always result in a longer ϵ-bounded network lifetime. For instance, $T/4$ provides the best improvement (around 16%) when ϵ is around 0.4, whereas $T/8$ provides the best improvement (around 40%) when ϵ is around 0.8.

Load Balance in Sampler Selection: Although saving battery life (energy) and increasing average lifetime of the network through the use of selective sampling is desirable, it is also important to make sure that the task of being a sampler node is equally distributed among the nodes. To illustrate the effectiveness of our sampler selection mechanism in achieving the goal of load balance, we compare the variation in the amount of time nodes have served as a sampler between our sampler selection scheme and a scenario where the sampler nodes are selected randomly. The improvement in the variance (i.e., the percentage of decrease in variance when using our approach compared to randomized approach) is plotted as a function of β for different σ values in Figure 9. For all settings, we observe an improvement above 50% provided by our sampler selection scheme.

8 Discussions

Setting of Selective Sampling Parameters – There are a number of system parameters involved in our selective sampling approach to data collection in sensor networks. Most notable are: α, τ_d, τ_c, τ_u, τ_f, and β. We have described various trade-offs involved in setting these parameters. We now give a general and somewhat intuitive guideline for a base configuration of these parameters. Among these parameters, τ_d is the one that is most straightforward to set. τ_d is the desired sampling period and defines the temporal resolution of the collected data. A default value of 1 seconds can provide more than enough temporal resolution for most environmental monitoring applications. When domain-specific data distance functions are used during the clustering phase, a basic guide for setting α is to set it to 1. This results in giving equal importance to data distance and hop distance factors. The clustering period τ_c and schedule update period τ_u should be set in terms of the desired sampling period τ_d. Other than the cases where the phenomenon of interest is highly dynamic, it is not necessary to perform clustering and schedule update frequently. However, re-clustering and re-assigning sampling schedules help achieve better load balancing due to alternated sampler nodes and cluster heads. As a result, one balanced setting for these parameters is $\tau_c = 1$ hour and $\tau_u = 15$ minutes. From our experimental results, we conclude that these values result in very little overhead. The forced sampling period τ_f defines the number of the sample readings used for calculating the probabilistic model parameters.

For the suggested setting of τ_u, having $\tau_f = 0.1 * \tau_u = 1.5$ minutes results in having 90 samples out of 900 readings per schedule update period. This is statistically good enough for calculating correlations. Finally, β is the subcluster granularity parameter and based on our experimental results, we suggest a reasonable setting of $\beta \in [5, 7]$. Note that both small and large values for beta will degrade the prediction quality.

Messaging Cost and Energy Consumption – One of the most energy consuming operations in sensor networks is the sending and receiving of messages, although the energy cost of keeping the radio in the active state also presents non-negligible cost [34]. It is important to note that our selective sampling approach to data collection operates in a completely periodic manner. By scheduling the selective sampling based data collection periodically, we ensure that during most of the time there is no messaging activity in the system. As a result, taking the number of messages exchanged during data collection as the major indicator of energy consumption is a sound assumption. Considering the fact that there exist generic protocols for exploiting such timing semantics found in most of the data collection applications [13], we can easily incorporate an exisiting energy efficient radio management protocol in order to save energy by reducing or avoiding the need for keeping the radio active during periods of inactivity.

Reliable Maintenance of Network Structures – There are three important network structures to be maintained in our selective sampling approach to data collection. These are the cluster head nodes, the cluster connection trees, and the data collection tree. Note that the cluster connection trees and the data collection tree are solely used for communicating with the cluster heads and the base node, respectively. This means that these trees need not be maintained in case there is a routing mechanism already existent in the network, such as geographical routing (ex. GPSR [25]) or other well-known ad-hoc routing mechanisms such as AODV, DSR, and DSDV (see [8]). The use of these trees in selective sampling is completely different than their use in aggregation schemes in which in-network processing is performed at each non-leaf node of the tree. In selective sampling, most of the in-network processing takes place in the cluster head nodes. As a result, there are two important issues that requires special attention for ensuring reliability. First, the failure of cluster head nodes should be detected and in case of failures new cluster head nodes should be elected. This can be achieved by applying classical primary/backup techniques from distributed computing. Second, it is important that the model parameters sent from the cluster head nodes are successfully transmitted to the base node. This can be achieved by employing end-to-end acknowledgment schemes between the cluster heads and the base node. Note that the loss of a message which includes a sensor reading destined to the base node is not a serious problem, since the value of such a node can be predicted at the base node using the probabilistic models and the readings of other nodes, albeit with some error. In comparison, the loss of a message in an aggregation scheme results in neglecting the values of all the nodes under a subtree.

9 Related Work

Energy efficiency plays a fundamental role in localized and distributed algorithms for wireless sensor networks in the context of various different services and protocols, such as broadcasting [24], message routing [40], medium access control [44], time synchronization [16], and location determination [37]. Data collection is another important service provided by sensor networks, especially for environmental monitoring applications. We review the literature related to sensor data collection in three categories: sensor data collection systems, node clustering in ad-hoc networks, and probabilistic inference in sensor networks.

Sensor Data Collection Systems – In Section 1 we have discussed the distinction between our selective sampling approach and other data collection approaches, such as those based on event detection [2], in-network aggregation [31], and distributed compression [36]. In summary, selective sampling is designed for energy efficient periodic collection of raw sensor readings from the network for the purpose of performing detailed data analysis that can not be done using in-network executed queries or locally detected events. The energy saving is a result of trading-off data accuracy, which is achieved by using a dynamically changing subset of nodes as samplers. This is in some ways similar to previously proposed energy saving sensor network topology formation algorithms, such as PEAS [43], where only a subset of nodes are made active, while preserving the network connectivity. Selective sampling uses a similar logic, but in a different context and for a different purpose: only a subset of nodes are used to actively sample, while the quality of the collected data is kept high.

There are a number of recent works [22, 15, 26] that has considered the trade-off between energy consumption and data collection quality. In [22] algorithms are proposed to minimize the sensor node energy consumption in the process of answering a set of user supplied queries with specified error thresholds. The queries are answered using uncertainty intervals cached at the server. These cached intervals are updated using an optimized schedule of server-initiated and sensor-initiated updates. Our selective sampling approach is not bound to queries and collects data periodically, so that both on-line and archival applications can make use of the collected data.

BBQ [15] is a model-driven data acquisition framework for sensor networks, that uses global optimizations for generating energy efficient data collection schedules. It is designed for multi-sensor systems, where nodes have multiple sensors with different energy consumption specifications. The correlations between readings from different sensors within a node are exploited to build statistical models, which enables prediction of the sensor readings that are energy-wise expensive to sample, from other sensor readings that are energy-wise cheap to sample, for instance temperature readings from voltage readings. Our selective sampling approach uses statistical models in a similar manner, but instead of modeling intra-node correlations among readings

of different-type sensors, we model inter-node correlations among the readings of same-type sensors from different nodes. Moreover, unlike [15], our framework is not query bound.

Snapshot Queries [26] is perhaps the most relevant work to ours. In [26], each sensor node is either represented by one of its neighbors or it is a representative node. Although this division is similar to sampler and non-sampler nodes of our selective sampling approach, there is a fundamental difference. The neighboring relationship imposed on representative nodes imply that the number of representatives is highly dependent on the connectivity graph of the network. For instance, as the connectivity graph gets sparse, the number of representative nodes may grow relative to the total network size. This restriction does not apply to the number of sampler nodes in selective sampling, since the selection process is supported by a clustering algorithm and is not limited to one-hop neighborhoods. In [26], representative nodes predict the values of their dependent neighbors for the purpose of query evaluation. This can cut down the energy consumption dramatically for aggregate queries, since a single value will be produced as an aggregate from the value of the representative node and the predicted values of the dependent neighbors. However this local prediction will not support such savings when queries have holistic aggregates [31] or require collection of readings from all nodes. Thus, selective sampling employs a hybrid approach where prediction is performed outside the network. Moreover, the model based prediction performed in our selective sampling approach uses correlation based schedule derivation to subcluster nodes into groups based on how good these nodes are in predicting each other's value. Any node within the same cluster can be put into the same subcluster, independent of the neighboring relationship between them. As opposed to this, snapshot queries does not use a model and instead employs binary linear regression for each representative-dependent node pair.

Node Clustering in Ad-hoc Networks – With respect to ad-hoc networks, most of the previous work in distributed node clustering have focused on constructing one-hop clusters [28, 4, 5, 7, 11]. In a one-hop cluster, each node is at most one-hop away from its cluster head. A few exceptions to this line of work are [12], [1], and [38]. In [1], a heuristic-based distributed algorithm is introduced for building clusters in which each node is at most d hops away from its cluster head. d is a system parameter and the algorithm tends to create clusterings in which clusters have approximately the same size. In [12], several distributed clustering algorithms are proposed for constructing k-hop clusters, where each node is at most k-hops away from the cluster head. In [38], a connectivity-based distributed node clustering algorithm is proposed, where nodes that are "highly connected" in the connectivity graph are put into the same cluster. All these algorithms, though distributed, do not attempt to cluster the network based on sensing structure. Hence the clusters discovered are not necessarily "good" clusters from a prediction stand-point. In contrast, our clustering algorithm is unique in being sensing-driven and results in clusters that improve prediction quality.

Inference in Sensor Networks – Our selective sampling approach to energy efficient data collection in sensor networks uses probabilistic models, whose parameters are locally inferred at the cluster head nodes and are later used at the base node to predict the values of non-sampler sensor nodes. Several recent works have also proposed to use probabilistic inference techniques to learn unknown variables within sensor networks [35, 20, 42, 9]. In [20], regression models are employed to fit a weighted combination of basis functions to the sensor field, so that a small set of regression parameters can be used to approximate the readings from the sensor nodes. In [9], probabilistic models representing the correlations between the sensor readings at various locations are used to perform distributed calibration. In [42], a distributed fusion scheme is described to infer a vector of hidden parameters that linearly relate to each sensor's reading with a Gaussian error. Finally, in [35] a generic architecture is presented to perform distributed inference in sensor networks. The solution employs message passing on distributed junction-trees, and can be applied to a variety of inference problems, such as sensor field modeling, sensor fusion, and optimal control.

10 Conclusion

We introduced *selective sampling* for energy-efficient periodic data collection in sensor networks. In particular, we showed that selective sampling can be effectively used to increase the network lifetime, while still keeping quality of the collected data high. We described three main mechanisms, (i) *sensing-driven cluster construction*, (ii) *correlation-based sampler selection and model derivation*, and (iii) *selective data collection and model-based prediction*, that together form the crux of our selective sampling approach. We demonstrated the effectiveness of selective sampling under different system settings through our reported results derived from analytical and simulation based experimental studies.

References

[1] A. D. Amis, R. Prakash, D. Huynh, and T. Vuong. Max-Min D-cluster formation in wireless ad hoc networks. In *IEEE INFOCOM*, 2000.

[2] C. I. ans Ramesh Govindan and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *ACM MobiCom*, 2000.

[3] T. Arici, B. Gedik, Y. Altunbasak, and L. Liu. PINCO: A pipelined in-network compression scheme for data collection in wireless sensor networks. In *IEEE ICCCN*, 2003.

[4] S. Bandyopadhyay and E. J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *IEEE INFOCOM*, 2003.

[5] S. Basagni. Distributed clustering for ad hoc networks. In *I-SPAN*, 1999.

[6] M. Batalin, M. Rahimi, Y. Yu, S. Liu, G. Sukhatme, and W. Kaiser. Call and response: Experiments in sampling the environment. In *ACM SenSys*, 2004.

[7] C. Bettstetter and R. Krausser. Scenario-based stability anlysis of the distributed mobility-adaptive clustering (DMAC) algorithm. In *ACM MobiHoc*, 2001.

[8] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *ACM MobiCom*, 1998.

[9] V. Byckovskiy, S. Megerian, D. Estrin, and M. Potkonjak. A collaborative approach to in-place sensor calibration. In *IEEE IPSN*, 2003.

[10] G. Casella and R. L. Berger. *Statistical Inference*. Duxbury Press, June 2001.

[11] M. Chatterjee, S. Das, and D. Turgut. Wca: A weighted clustering algorithm for mobile ad hoc networks. *Journal of Cluster Computing*, 5, April 2002.

[12] G. Chen and I. Stojmenovic. Clustering and routing in mobile wireless networks. Technical report, SITE, University of Ottawa, 1999.

[13] O. Chipara, C. Lu, and G.-C. Roman. Efficient power management based on application timing semantics for wireless sensor networks. In *IEEE ICDCS*, 2005.

[14] Crossbow technology. http://www.xbow.com, December 2004.

[15] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, 2004.

[16] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *IEEE IPDPS*, 2001.

[17] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1, January 2002.

[18] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *ACM MobiCom*, 1999.

[19] Global precipitation climatology project. http://www.ncdc.noaa.gov/oa/wmo/wdcamet-ncdc.html, December 2004.

[20] C. Guestrin, R. Thibaux, P. Bodik, M. A. Paskin, and S. Madden. Distributed regression: An efficient framework for modeling sensor network data. In *IEEE IPSN*, 2004.

[21] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, August 2000.

[22] Q. Han, S. Mehrotra, and N. Venkatasubramanian. Energy efficient data collection in distributed sensor environments. In *IEEE ICDCS*, 2004.

[23] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *ACM ASPLOS*, 2000.

[24] I.Stojmenovic, M. Seddigh, and J. Zunic. Dominating sets and neighbor elimination-based broadcasting

algorithms in wireless networks. *IEEE TPDS*, 13(1), 2002.

[25] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *ACM MobiCom*, 2000.

[26] Y. Kotidis. Snapshot queries: Towards data-centric sensor networks. In *IEEE ICDE*, 2005.

[27] D. Li, K.Wong, Y. Hu, and A. Sayeed. Detection, classification, tracking of targets in micro-sensor networks. *IEEE Signal Processing Magazine*, March 2002.

[28] C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 15(7), 1997.

[29] J. Liu, J. Reich, P. Cheung, and F. Zhao. Distributed group management for track initiation and maintenance in target localization applications. In *Workshop on Information Processing in Sensor Networks*, 2003.

[30] L. Liu, C. Pu, and W. Tang. Continual queries for internet scale event-driven information delivery. *IEEE TKDE*, 11, July/August 1999.

[31] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *USENIX OSDI*, 2002.

[32] S. Madden, R. Szewczyk, M. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *IEEE Workshop on Mobile Computing Systems and Applications*, 2002.

[33] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[34] Moteiv. telos revb datasheet. http://www.moteiv.com/pr/2004-12-09-telosb.php, December 2004.

[35] M. A. Paskin and C. E. Guestrin. A robust architecture for distributed inference in sensor networks. In *IEEE IPSN*, 2005.

[36] S. Pradhan, J. Kusuma, and K. Ramchandran. Distributed compression in a dense sensor network. *IEEE Signal Processing Magazine*, March 2002.

[37] N. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Anchor-free distributed localization in sensor networks. Technical report, MIT Laboratory for Computer Science, 2003.

[38] L. Ramaswamy, B. Gedik, and L. Liu. Connectivity based node clustering in decentralized peer-to-peer networks. In *IEEE P2P*, 2003.

[39] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akylidiz. ESRT: Event-to-sink reliable transport in wireless sensor networks. In *ACM MobiHoc*, 2003.

[40] C. Schurgers and M. B. Srivastava. Energy efficient routing in wireless sensor networks. In *MILCOM*, 2001.

[41] Taos Inc. ambient light sensor (ALS). http://www.taosinc.com/images/product/document/tsl2550-e58.pdf, December 2004.

[42] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *IEEE IPSN*, 2005.

[43] F. Ye, G. Zhong, S. Lu, and L. Zhang. Peas: A robust energy conserving protocol for long-lived sensor networks. In *IEEE ICDCS*, 2003.

[44] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *IEEE INFOCOM*, 2002.