ASAP: An Adaptive Sampling Approach to Data Collection in Sensor Networks

Buğra Gedik, *Member*, *IEEE*, Ling Liu, *Senior Member*, *IEEE*, and Philip S. Yu, *Fellow*, *IEEE*

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 develop ASAP, which is an adaptive sampling approach to energy-efficient periodic data collection in sensor networks. The main idea behind ASAP is to use a dynamically changing subset of the nodes as samplers such that the sensor readings of the sampler nodes are directly collected, whereas the values of the nonsampler nodes are predicted through the use of probabilistic models that are locally and periodically constructed. ASAP can be effectively used to increase the network lifetime while keeping the quality of the collected data high in scenarios where either the spatial density of the network deployment is superfluous, which is relative to the required spatial resolution for data analysis, or certain amount of data quality can be traded off in order to decrease the power consumption of the network. The ASAP 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* are used to determine the sampler nodes and to calculate the parameters of the probabilistic models that capture the spatial and temporal correlations among the sensor readings. Last, *adaptive data collection and model-based prediction* are used to minimize the number of messages used to extract data from the network. A unique feature of ASAP is the use of in-network schemes, as opposed to the protocols requiring centralized control, to select and dynamically refine the subset of the sensor nodes serving as samplers and to adjust the value prediction models used for nonsampler nodes. Such runtime adaptations create a data collection schedule, which is self-optimizing in response to the changes in the energy levels of the nodes and environmental dynamics. We present simulation-based experimental results and study the effectiveness of ASAP under different system settings.

Index Terms—Sensor networks, data communications, data models.

1 INTRODUCTION

The proliferation of low-cost tiny sensor devices (such as the Berkeley Mote [1]) and their 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 [2], [3]. On the downside, the large and growing number of networked sensors present a number of unique system design challenges, which are 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 [4], depending on the particular type of sensing performed (for example, solar radiation sensors [5]). On the other hand, computation is relatively less energy consuming.
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 the environmental dynamics, changes in the user demands that contribute to the task dynamics as to what is being sensed and what is considered interesting changes [6], and changes in the energy levels, location, or connectivity of the sensor nodes that contribute to the network dynamics.

One of the main objectives in configuring networks of sensors for large-scale data collection is to achieve longer lifetimes for the sensor network deployments by keeping the energy consumption at the minimum while maintaining sufficiently high quality and resolution of the collected data to enable a meaningful analysis. Furthermore, the configuration of data collection should be readjusted from time to time in order to adapt to changes resulting from high system dynamics.

1.1 Data Collection in Sensor Networks

We can broadly divide data collection, which is a major functionality supported by sensor networks, into two categories. In *event-based* data collection (for example, REED

- *B. Gedik and P.S. Yu are with the IBM Thomas J. Watson Research Center, 19 Skyline Dr., Hawthorne, NY 10532. E-mail: {bgedik, psyu}@us.ibm.com.*
- *L. Liu is with the College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30332. E-mail: lingliu@cc.gatech.edu.*

Manuscript received 10 Aug. 2006; revised 5 Feb. 2007; accepted 10 Apr. 2007; published online 8 May 2007.
Recommended for acceptance by C. Raghavendra.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0224-0806.
Digital Object Identifier no. 10.1109/TPDS.2007.1110.

[7]), the sensors are responsible for detecting and reporting (to a base node) events such as spotting moving targets [8]. The 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 nontrivial distributed algorithms [9] that require the involvement of multiple sensor nodes. An inherent downside of the event-based data collection is the impossibility of performing an in-depth analysis on the raw sensor readings since they are not extracted from the network in the first place.

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 [10]) 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 [11], [12] support aggregate queries such as minimum, average, and maximum. These types of queries result in periodically generating an aggregate of the recent readings of all nodes. Although aggregation lends itself to simple implementations that enable the complete in-network processing of queries, it falls short in supporting holistic aggregates [11] over sensor readings such as quantiles. Similar to the case of event-based data collection, the raw data is not extracted from the network and a complex data analysis that requires the integration of sensor readings from various nodes at various times cannot be performed with the in-network aggregation.

The most comprehensive way of data collection is to extract sensor readings from the network through the periodic reporting of each sensed value from every sensor node.[1] This scheme enables arbitrary data analysis at a sensor stream processing center once the data is collected. Such an increased flexibility in data analysis comes at the cost of high energy consumption due to excessive communication and consequently decreases the network lifetime. In this paper, we develop ASAP, which is an adaptive sampling approach to energy-efficient periodic data collection. The main idea behind ASAP is to use a carefully selected dynamically changing subset of nodes (called *sampler* nodes) to sense and report their values and to predict the values of the rest of the nodes by using *probabilistic models*. Such models are constructed by exploiting both spatial and temporal correlations that are existent among the readings of the sensor nodes. Importantly, these models are locally constructed and revised in order to adapt to the changing system dynamics.

1.2 Perspective Place in Related Literature

Before presenting the contributions of this paper, we will first put our work into perspective with respect to previous work in the area of model-based data collection.

Model-based data collection is a technique commonly applied to reduce the amount of communication required to collect sensor readings from the network or to optimize the sensing schedules of nodes to save energy. Examples include the Barbie-Q: A Tiny-Model Query System (BBQ) [4] and Ken [13] sensor data acquisition systems. There are two main processes involved in model-based data collection. These are probabilistic model construction and model-based value prediction. However, different approaches differ in terms of the kinds of correlations in sensor readings that are being modeled, as well as with regard to where in the system and how the model construction and prediction are performed. According to the first criterion, we can divide the previous work in this area into two categories: *intranode* modeling and *internode* modeling.

In intranode modeling, correlations among the readings of different type sensors on the same node are modeled, such as the correlations between the voltage and temperature readings within a multisensor node. For instance, BBQ [4] models these intranode correlations and creates optimized sensing schedules in which low-cost sensor readings are used to predict the values of the high-cost sensor readings. The strong correlations among the readings of certain sensor types enable accurate prediction in BBQ. In the context of intranode correlations, model-based prediction support for databases [14] and attribute correlation-aware query processing techniques [15] have also been studied.

In internode modeling, correlations among readings of the same type of sensors on different but spatially close-by nodes are modeled. For instance, Ken [13] uses the spatial and temporal correlations among the sensor node readings to build a set of probabilistic models, one for each correlated cluster of nodes. Such models are used by the cluster heads within the sensor network to suppress some of the sensor readings that can be predicted at the server side by using the constructed models, achieving data collection with low messaging overhead but satisfactory accuracy.

In this paper, our focus is on internode modeling, which can also be divided into two subcategories based on where in the system the probabilistic models are constructed. In the *centralized* model construction, the probabilistic models and the node clusters corresponding to these probabilistic models are constructed on the server or base node side, whereas, in the *localized* model construction, the models are locally discovered within the sensor network in a decentralized manner. Ken [13] takes the former approach. It uses historical data from the network to create clusters, select cluster heads, and build probabilistic models. This approach is suitable for sensor network deployments with low levels of system dynamics since the probabilistic models are not updated once they are built.

ASAP promotes the in-network construction of probabilistic models and clusters. There are two strong motivations for this. First, when the environmental dynamics are high, the probabilistic models must be revised or reconstructed. A centralized approach will introduce significant messaging overhead for keeping the probabilistic models up to date, as it will later be proven in this paper. Second, the energy levels of nodes will likely differ significantly due to the extra work assigned to the cluster head nodes. As a result, the clusters may need to be reconstructed to balance

1. Sometimes referred to as SELECT * queries [13].

TABLE 1
Notations for Network Architecture

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 2
Notations for Sensing-Driven Clustering

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 sensed
T_i	Smallest hop distances from the cluster heads in proximity of p_i, as known to p_i during cluster formation
V_i	Sensor reading means of the cluster heads in proximity of p_i, as known to p_i during cluster formation
Z_i	Attraction scores for the 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 3
Notations for Correlation-Based
Sampler Selection and Model Derivation

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 4
Notations for Adaptive Data Collection
and Model-Based Prediction Parameters

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 $G_i(j)$. $\mathcal{Y}_{i,j}[p_u, p_v]$ 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

the energy levels of nodes so that we can keep the network connectivity high and achieve a longer network lifetime.

Localized approaches to model construction are advantageous in terms of providing self-configuring and self-optimizing capabilities with respect to environmental dynamics and network dynamics. However, they also create a number of challenges due to decentralized control and lack of global knowledge. In particular, ASAP needs decentralized protocols for creating clusters to facilitate local control and utilize the local control provided by these clusters to automatically discover a set of subclusters that are formed by nodes sensing spatially and temporally correlated values, making it possible to construct probabilistic models in an in-network and localized manner. ASAP achieves this through a three-phase framework that employs localized algorithms for generating and executing energy-aware data collection schedules.

The rest of this paper is organized as follows: In Section 2, we give a basic description of our system architecture and provide an overview of the algorithms involved in performing adaptive sampling. In Sections 3, 4, and 5, we describe the main components of our solution in turn, namely, 1) *sensing-driven cluster construction*, 2) *correlation-based sampler selection and model derivation*, and 3) *adaptive data collection with model-based prediction*. Discussions are provided in Section 6. In Section 7, we present several results from our performance study. We compare our work with the related work in Section 8, and we conclude in Section 9.

2 System Model and Overview

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

2.1 Network Architecture

We design our adaptive sampling-based data collection framework by 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 nodes that can communicate with each other form a *connectivity graph*. Fig. 1 depicts a segment from a network of 100 sensor nodes. The edges of the connectivity graph are shown with light gray lines. 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 Fig. 1, the base node is the shaded one labeled as "56." The edges of the data collection tree are shown in dark gray in Fig. 1. The data collection tree can be easily built in a distributed manner, for instance, by circulating a tree formation message that originated at the base node and making use

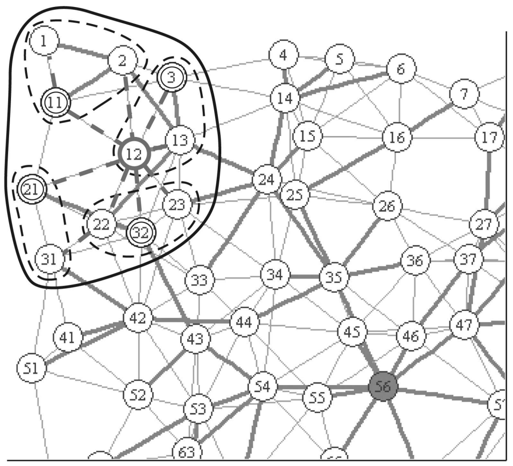
Fig. 1. Network architecture.

of a min-hop parent selection policy [16] or similar algorithms used for the in-network aggregation [12], [11].

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. There is also a *cluster-connection tree* with the cluster head as its root node, which is used to establish the communication between the cluster head and the other nodes in the cluster. We associate each node p_i with a cluster head indicator h_i, $i \in \{1, \dots, N\}$, 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 that has p_i as its cluster head ($h_j = p_i$), we say that p_j is in C_i ($p_j \in C_i$). A cluster is illustrated on the upper left corner of Fig. 1, with a closed line covering the nodes that belong to the cluster. The cluster head node is drawn in boldface 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).

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 $G_i(j)$, $j \in \{1, \dots, K_i\}$, 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. 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 Fig. 1, we show four subclusters for the node cluster of head node "12." The subclusters are circled with closed dashed lines in the figure.

A key feature of ASAP is that not all of the nodes in the network need to sense and report their 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 this subcluster. From now on, we refer to the nodes that sense and report to the base node as *sampler* nodes. In Fig. 1, the sampler nodes are marked with double circled lines (that is, 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 the node $p_j \in C_i$ is a sampler or not. We use the [] notation whenever the indexing is by nodes.

2.2 Adaptive-Sampling-Based Data Collection Overview

We give an overview of the three main mechanisms that form the crux of the ASAP approach. Detailed descriptions of these mechanisms are 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, which is executed periodically 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 we can choose a 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, in 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 early. 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 sensor readings in the past should be clustered into one group. Second, nodes that are clustered together should be close to each other in terms of network hops. The first metric is based on the value similarity of sensor readings, which is a distinguishing feature compared to the naive minimum-hop-based cluster formation, where a node joins the cluster that has the closest cluster head 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 prediction of nonsampler 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*. 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 infrequently sampled readings from *all* the nodes within its cluster to capture the spatial and temporal correlations in sensor readings and calculate the subclusters so that the nodes whose sensor readings 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 systemwide 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 report sensor readings to the base node and the values of the nonsampler nodes will be predicted at the processing center (or the base node) by using probabilistic models that are constructed in an in-network manner for each subcluster and reported to the base node. Thus, the third step here is to construct and report a probabilistic model for each subcluster within the network. 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 sensed values from the network and to perform the prediction after the sensor readings are received. This is achieved by the *adaptive data collection and model-based prediction* algorithm. The adaptive data collection component works in two steps: 1) Each sampler node senses a value every τ_d seconds, called the *desired sampling period*. τ_d sets the temporal resolution of the data collection. 2) To empower ASAP with self-adaptation, we also need to periodically but infrequently collect (at the cluster heads) sensor readings from all nodes within a cluster. Concretely, at every τ_f seconds (τ_f is a multiple of τ_d), all nodes perform sensing. These readings are collected (through the use of cluster-connection trees) and used by the cluster head nodes, aiming at incorporating the newly established correlations among the sensor readings into the decision-making process of the correlation-based sampler selection and model derivation algorithm. τ_f is a system-supplied parameter, called the *forced sampling period*. The model-based prediction component is responsible for estimating the values of the nonsampler nodes within each subcluster by using the readings of the sampler nodes and the parameters of the probabilistic models constructed for each subcluster.

3 Sensing-Driven Cluster Construction

The goal of sensing-driven cluster construction is to form a network organization that can facilitate adaptive 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 an in-network manner. By emphasizing on the sensing-driven clustering, it also helps to derive better prediction models to increase the prediction quality.

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 at the beginning of a cluster head selection phase. A node p_i first calculates a value called the *head selection probability*, denoted by s_i. This probability is calculated based on two factors. The first one is a systemwide parameter called the *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. 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 $e_i(t)$. 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 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 the most realistic scenarios governing energy values that are available at nodes and the practical settings of f_c (< 0.2), this process results in selecting $\approx f_c * N$ cluster heads.

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*.

3.2.1 Message Circulation

This 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 p_i. This field represents the originator of the cluster formation message. It sets $m.ttl$ to TTL, which is a systemwide parameter that defines the maximum number of hops that this message can travel within the network. This field indicates the number of remaining hops that 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 that node p_i has sensed 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. If $m.rnd$ is larger than r_i, then this is the first cluster formation message that 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 the dmu field of the cluster formation message that originated at the head node p_j and reached p_i. Once the processing of the rnd field of the message is over, we calculate the number of hops that 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 that this is not the first message that we received in this round, which originated at node $m.org$), and $T[m.org]$ is smaller than or equal to the number of hops that 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 the src field of the message. The modification on the message involves decrementing the ttl field and setting the src field to p_i.

3.2.2 Cluster Engagement

This step involves making a decision about which cluster to join once the information on hop distances and mean sample values 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 (that is, $\{p_j | T_i[p_j] \neq \emptyset\}$), 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; that is, 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: *hop distance factor* and *data distance factor*.

The hop distance factor 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 is $1 - 1/TTL$ when p_i is one hop away from p_j. The data distance factor 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 sensed 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 it approaches 0 when the difference approaches 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 systemwide 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 $max(0, 1 - f(d))$.

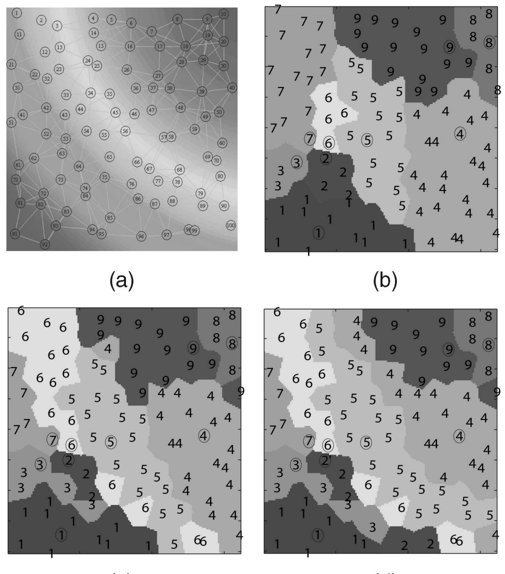

Fig. 2. Sensing-driven clustering.

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 the *data importance factor*, denoted by α. α takes a value in the range $[0, \infty)$. A value of 0 means that 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.

3.3 Effect of α on Clustering

We use an example scenario to illustrate the effect of α on clustering. Fig. 2a 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 light gray to dark gray moving diagonally from the upper right corner to the lower left corner, representing a decrease in the sensed values. Figs. 2b, 2c, and 2d show three different clusterings of the network for different values of α (0, 10, and 20, respectively) by 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 an 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 than TTL hops away. We provide a quantitative study on the effect of α on the quality of the clustering in Section 7.

In Figs. 2c and 2d, one can observe that, by combining the hop distance factor and sensor reading similarity captured by the 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 cannot be large in practice due to three major reasons: First, since there is a fixed TTL value that is used in cluster creation, the nodes that belong to the same cluster cannot 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 that 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 the hop distance factor and the 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 will not 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 the 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 (discussed 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 the τ_c parameter should be small enough to enable the selection of alternate nodes as cluster heads. However, its value is expected to be much larger than the desired sampling and forced sampling periods, that is, τ_d and τ_f. Second, time-dependent changes in sensor readings may render the current clustering obsolete with respect to the data distance factor. For instance, in environmental monitoring applications, different times of day may result in different node clusters. Thus, the clustering period should be adjusted accordingly to enable continued refinement of the clustering structure in response to different sensing patterns resulting from environmental changes.

4 Correlation-Based Sampler Selection and Model Derivation

The goal of sampler selection and model derivation is threefold. 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 nodes within the cluster) a sampling schedule that defines the sampler nodes. Third, it needs to derive and report (to the base node) the parameters of the probabilistic models associated with each subcluster so that value prediction for nonsampler nodes 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 steps in detail.

4.1 Subclustering

Higher correlations among the nodes within a subcluster typically lead to higher quality sampler selection and higher accuracy in the model-based value prediction of nonsampler nodes. Thus, given a cluster, the first issue involved in developing an effective subclustering is to obtain sensor readings from the nodes within this cluster. The second issue is to compute correlations between nodes within the cluster and define a correlation distance metric that can be used as the distance function for subclustering.

4.1.1 *Forced Sampling*

Recall from Section 2.2 that we have introduced the concept of forced sampling period. By periodically collecting sensor readings from all nodes within a cluster (forced sampling), the cluster head nodes can refine the subclustering structure when a new run of the subclustering process is started. Subclustering 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 column vector of consecutive forced samples from node p_j, where p_j is in the node cluster, with p_i as the head (that is, $p_j \in C_i$).

4.1.2 *Correlation Matrix and 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 [17] of the correlation between two series, expressed using the notations introduced within the context of this work. The correlation values are always in the range $[-1, 1]$, with -1 and 1 representing the strongest negative and positive correlations. A value of 0 implies that 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 sensor reading from another's. For each node cluster, we first compute its correlation matrix by using forced samples. Then, we calculate the correlation distance matrix 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 [18] to subcluster the nodes within cluster C_i into K_i number of subclusters, where $G_i(j)$ denotes the set of nodes in the jth subcluster. We use a systemwide parameter called *subcluster granularity*, denoted by β, to define the average subcluster size. Thus, K_i is calculated by $\lceil |C_i|/\beta \rceil$. We will discuss the effects of β on performance later in this section. The pseudocode for the subclustering step is given within the SUBCLUSTERANDDERIVE procedure in Algorithm 1.

Algorithm 1: Correlation-based sampler selection and model derivation.

```
DERIVESCHEDULE($p_i \in H$)
(1)  Periodically, every $\tau_u$ seconds
(2)      $D_i$: data collected since last schedule derivation,
         $D_i[p_j](k)$ is the $k$th forced sample from node $p_j$
         collected at node $p_i$
(3)      $(S_i, \mathcal{C}_i, G_i) \leftarrow$ SUBCLUSTERANDDERIVE($p_i$, $D_i$)
(4)      for $j \leftarrow 1$ to $|G_i|$
(5)          $\mathcal{X}_{i,j} : \mathcal{X}_{i,j}[p_u] = \mathrm{E}[D_i[p_u]]; p_u \in G_i(j)$
(6)          $\mathcal{Y}_{i,j} : \mathcal{Y}_{i,j}[p_u, p_v] = \mathcal{C}_i[p_u, p_v] * \sqrt{\mathrm{Var}(D_i[p_u])} *$
             $\sqrt{\mathrm{Var}(D_i[p_v])}$; $p_u$, $p_v \in G_i(j)$, $u < v$
(7)          SENDMSG($base, \mathcal{X}_{i,j}, \mathcal{Y}_{i,j}$)
(8)      foreach $p_j \in C_i$
(9)          $D_i[p_j] \leftarrow \emptyset$
(10)         SENDMSG($p_j, S_i[p_j]$)

SUBCLUSTERANDDERIVE($p_i \in H$)
(1)  $\forall p_u$, $p_v \in C_i$, $\mathcal{C}_i[p_u, p_v] \leftarrow$ Correlation between $D_i[p_u], D_i[p_v]$
(2)  $\forall p_u$, $p_v \in C_i$, $\mathcal{D}_i[p_u, p_v] \leftarrow 1 - |\mathcal{C}_i[p_u, p_v]|$
(3)  $K_i \leftarrow \lceil |C_i|/\beta \rceil$  /* number of subclusters */
(4)  Cluster the nodes in $C_i$, using $\mathcal{D}_i$ as distance metric,
     into $K_i$ subclusters
(5)  $G_i(j)$: nodes in the $j$th subcluster within $C_i$,
     $j \in \{1, \dots, K_i\}$
(6)  $t \leftarrow$ Current time
(7)  $\forall p_u \in C_i$, $S_i[p_u] \leftarrow 0$
(8)  for $j \leftarrow 1$ to $K_i$
(9)      $a \leftarrow \lceil \sigma * |G_i(j)| \rceil$
(10)     foreach $p_u \in G_i(j)$, in decreasing order of $e_u(t)$
(11)         $S_i[p_u] \leftarrow 1$
(12)         if $a = |\{p_v | S_i[p_u] = 1\}|$ then break
(13) return $(S_i, \mathcal{C}_i, G_i)$
```

4.2 Sampler Selection

This 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, that is, $S_i[p_j] = 0$, $\forall p_j \in C_i$. 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. Nodes with more energy remaining are preferred as samplers within a subcluster, and at least one node is selected ($S_i[p_j] = 1$ if p_j is selected) as a sampler node from each subcluster. Concretely, we calculate the number of sampler nodes for a given subcluster $G_i(j)$ by $\lceil \sigma * |G_i(j)| \rceil$. Based on this formula, we can derive the actual fraction of the nodes selected as samplers. 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 pseudocode for the derivation step is given within the SUBCLUSTERANDDERIVE procedure in Algorithm 1.

4.3 Model and Schedule Reporting

This 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 nonsamplers. Then, the cluster head sends the summary information to the base node, which will be used to derive the parameters of the probabilistic models used in predicting the values of nonsampler 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 the nodes within the subcluster, denoted by $\mathcal{X}_{i,j}$, as $\mathcal{X}_{i,j}[p_u] = \mathrm{E}[D_i[p_u]]$, $p_u \in G_i(j)$. p_i also calculates a data covariance matrix for the nodes within the subcluster, denoted by $\mathcal{Y}_{i,j}$ and defined as

$$\mathcal{Y}_{i,j}[p_u, p_v] = \mathcal{C}_i[p_u, p_v] * \sqrt{\mathrm{Var}(D_i[p_u])} * \sqrt{\mathrm{Var}(D_i[p_v])},$$
$$p_u,\ p_v \in G_i(j).$$

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 Multivariate Normal (MVN) model for each subcluster (see Section 5). The pseudocode is given as part of the DERIVESCHEDULE procedure of Algorithm 1.

4.4 Effects of β on Performance

The setting of the system-supplied parameter β (subcluster granularity) affects the overall performance of an ASAP-based data collection system, especially in terms of sampler selection quality, value prediction 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, values that are too small will decrease the prediction quality since the opportunity to exploit the spatial correlations fully will be missed with very small β values. Regarding the messaging cost of sending the 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 the covariance values of node pairs belonging to different subclusters will not be reported. Although the second dimension favors a small β value, decreasing β will increase the deviation of effective σ from the system-specified σ, which introduces 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 that is 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 our experimental evaluation (see Section 7).

4.5 Setting of the Schedule Update Period τ_u

The schedule update period τ_u is a system-supplied parameter and it defines the time interval for recomputing 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 nonsamplers since they perform sensing and report their sensed values. Consequently, the value of the τ_u parameter should be small enough to enable selection of alternate nodes as samplers through the use of the energy-aware schedule derivation process in order to balance the 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 reselection 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 the dynamics of the application at hand. If the rate at which the correlations among the sensors change is extremely high for an application, then the subclustering step may need to be repeated frequently (small τ_u). This can potentially incur high messaging overhead. However, it is important to note that ASAP still incurs less overhead compared to centralized model-based prediction frameworks since the model parameters are recomputed within the network in a localized manner.

5 Adaptive Data Collection and Model-Based Prediction

ASAP achieves energy efficiency of data collection services by collecting sensor readings from only a subset of nodes (sampler nodes) that are carefully selected and are dynamically changing (after every schedule update period τ_u). The values of nonsampler nodes are predicted using probabilistic models whose parameters are derived from the recent readings of nodes that are spatially and temporally correlated. The energy saving is a result of a smaller number of messages used to extract and collect data from the network, which is a direct benefit of a smaller number of sensing operations performed. Although all nodes have to sense after every forced sampling period (recall that these readings are used for predicting the parameters of the 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.

In effect, one sensor value for each node is calculated at the base node (or at the sensor stream processing center). However, a sensor value comes from either a *direct* sensor reading or a *predicted* sensor reading. If a node p_i is a sampler, that is, $S_j[p_i] = 1$, where $h_i = p_j$, it periodically reports its sensor reading to the base node by using the data collection tree, that is, after every desired sampling period τ_d, except when the 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 by using the cluster-connection tree and is forwarded to the base node from there. If a node p_i is a nonsampler node, that is, $S_j[p_i] = 0$ 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 by using the cluster-connection tree and are *not* forwarded to the base node. The pseudocode for this is given by the SENSDATA procedure in Algorithm 2.

Algorithm 2: Adaptive data collection and model-based prediction.

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

PREDICTDATA($i, j, U^+, U^-, W^+$)
$U^+ = \{p_{u_1^+}, \dots, p_{u_k^+}\}$: set of sampler nodes in $G_i(j)$
$U^- = \{p_{u_1^-}, \dots, p_{u_l^-}\}$ : set of nonsampler nodes in $G_i(j)$
$W^+ : W^+(a), a \in \{1, \dots, k\}$ is the value reported by node $p_{u_a^+}$
(1)  $\mathcal{X}_{i,j}$: mean vector for $G_i(j)$
(2)  $\mathcal{Y}_{i,j}$: covariance matrix for $G_i(j)$
(3)  for $a \leftarrow 1$ to $l$
(4)      $\mu_1(a) \leftarrow \mathcal{X}_{i,j}[p_{u_a^-}]$
(5)      for $b \leftarrow 1$ to $l$, $\Sigma_{11}(a,b) \leftarrow \mathcal{Y}_{i,j}[p_{u_a^-}, p_{u_b^-}]$
(6)      for $b \leftarrow 1$ to $k$, $\Sigma_{12}(a,b) \leftarrow \mathcal{Y}_{i,j}[p_{u_a^-}, p_{u_b^+}]$
(7)  for $a \leftarrow 1$ to $k$
(8)      $\mu_2(a) \leftarrow \mathcal{X}_{i,j}[p_{u_a^+}]$
(9)      for $b =\leftarrow 1$ to $k$, $\Sigma_{22}(a,b) \leftarrow \mathcal{Y}_{i,j}[p_{u_a^+}, p_{u_b^+}]$
(10) $\mu^* = \mu_1 + \Sigma_{12} * \Sigma_{22}^{-1} * (W^+ - \mu_2)$
(11) $\Sigma^* = \Sigma_{11} - \Sigma_{12} * \Sigma_{22}^{-1} * \Sigma_{12}^T$
(12) Use $\mathcal{N}(\mu^*, \Sigma^*)$ to predict values of nodes in $U^-$
```

5.1 Calculating Predicted Sensor Values

The problem of predicting the sensor values of nonsampler nodes can be described as follows: Given a set of sensor values belonging to the same sampling step from sampler nodes within a subcluster $G_i(j)$, how can we predict the set of values belonging to the nonsampler 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 nonsampler 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 sensor values from the same sampling step, which is received from the sampler nodes in $U^+_{i,j}$. Using an MVN model to capture the spatial and temporal correlations within a subcluster, we utilize the following theorem from statistical inference [17] to predict the values of the nonsampler nodes:

Theorem 1. *Let X be an 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 be 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 an 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 Σ_{11}, Σ_{12}, Σ_{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 nonsampler 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 this paper. The details are given by the PREDICTDATA procedure in Algorithm 2.

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 MVN 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 a Bayesian inference. This can provide significant improvement in prediction quality if prior distributions of the sensor readings 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 compact in size.

5.3 Setting of the Forced and Desired Sampling Periods: τ_f and τ_d

The setting of the forced sampling period τ_f involves three considerations. First, an increased number of forced samples (thus, smaller τ_f) is desirable, since it can improve the ability to capture correlations in sensor readings better. Second, a large number of forced samples can cause the memory constraint on the 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 occupies R units, then we should set τ_f to a value larger than $\frac{\tau_u * R}{f_c * M}$. Third, a less frequent forced sampling results in a 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 the desired sampling period τ_d defines the temporal resolution of the collected data and is application specific.

6 Discussions

In this section, we discuss a number of issues related with our adaptive sampling-based approach to data collection in sensor networks.

Setting of the ASAP *Parameters*. There are a number of system parameters involved in ASAP. The 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. It should be specified by the 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 an equal importance to the 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, reclustering and reassigning 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, assuming $\tau_d = 1$ seconds. 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 sensor readings used for calculating the probabilistic model parameters. For the suggested setting of τ_u, having $\tau_f = 0.5$ minutes results in having 30 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 β 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 incurs nonnegligible cost [19]. The latter is especially a major factor when the desired sampling period is large and, thus, the listening of the radio dominates the cost in terms of energy consumption. Fortunately, such large sampling periods also imply great opportunities for saving energy by turning off the radio. It is important to note that ASAP operates in a completely periodic manner. Most of the time, there is no messaging activity in the system since all nodes sample around the same time. As a result, application-level power management can be applied to solve the idle listening problem. There also exist generic power management protocols for exploiting the timing semantics of data collection applications [20]. For smaller desired sampling periods, energy-saving media access control (MAC) protocols like sensor-MAC (S-MAC) [21] can also be employed in ASAP (see Section 2.2).

Practical Considerations for Applicability. There are two major scenarios in which the application of ASAP in periodic data collection tasks is very effective. First, in scenarios where the spatial resolution of the deployment is higher than the spatial resolution of the phenomena of interest, ASAP can effectively increase the network lifetime. The difference in the spatial resolution of the deployment and the phenomena of interest can happen due to several factors, including 1) high cost of redeployment due to harsh environmental conditions, 2) different levels of interest in the observed phenomena at different times, and 3) different levels of spatial resolution of the phenomena at different times (for example, biological systems show different levels of activity during the day and night). Second, ASAP can be used to reduce the messaging and sensing cost of data collection when a small reduction in the data accuracy is acceptable in return for a longer network lifetime. This is achieved by exploiting the strong spatial and temporal correlations that are existent in many sensor applications. The ASAP solution is most appropriate when these correlations are strong. For instance, environmental monitoring is a good candidate application for ASAP, whereas anomaly detection is not.

7 Performance Study

We present simulation-based experimental results to study the effectiveness of ASAP. We divided the experiments into three sets. The first set of experiments studies the performance with regard to messaging cost. The second set of experiments studies the performance from the energy consumption perspective (using results from the ns2 network simulator). The third set of experiments studies the quality of the collected data (using a real-world data set). Further details about the experimental setup are given in Sections 7.1, 7.2, and 7.3.

For the purpose of comparison, we introduce two variants of ASAP: the *central* approach and the *local* approach. The central approach presents one extreme of the spectrum, in which both the model prediction and the value prediction of nonsampling 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 centrally. In the local approach, value prediction is performed at the cluster heads instead of the processing center outside the network, and predicted values are reported to the base node. The ASAP solution falls in between these two extremes. We call it the *hybrid* approach due to the fact that the spatial/temporal correlations are summarized locally within the network, whereas the value prediction is performed centrally at the base node. We refer to the *naive* periodic data collection with no support for adaptive sampling as the *no-sampling* approach.

7.1 Messaging Cost

Messaging cost is defined as the total number of messages sent in the network for performing data collection. We compare the messaging cost for different values of the system parameters. In general, the gap between the central and hybrid approaches, with the hybrid being less expensive and, thus, better, indicates the savings obtained by only reporting the summary of the correlations (hybrid approach) instead of forwarding the forced samples up to the base node (central approach). On the other hand, the gap between the local and no-sampling approaches, with the local approach being more expensive, indicates the overhead of cluster construction, sampler selection and model derivation, and adaptive data collection steps. Note that the local approach does not have any savings due to adaptive sampling since a value (direct or predicted reading) is reported for all nodes in the system.

The default parameters used in this set of experiments are given as follows: The total time is set to $T = 10^6$ units. The total number of nodes in the network is set to $N = 600$ unless specified otherwise. f_c is selected to result in an average cluster size of 30 nodes. The desired and forced sampling periods are set to $\tau_d = 1$ and $\tau_f = 10$ time units, respectively. The schedule update period is set to $\tau_u = 900$ time units and the clustering period is set to $\tau_c = 3{,}600$ time units. The sampling fraction is set to $\sigma = 0.25$ and the subcluster granularity parameter is set to $\beta = 10$. The nodes are randomly placed in a 100 m × 100 m grid and the communication radius of the nodes is taken as 5 m.

7.1.1 Effect of the Sampling Fraction σ

Fig. 3a plots the total number of messages as a function of the sampling fraction σ. We make several observations from the figure. First, the central and hybrid approaches provide significant improvement over the local and no-sampling approaches. This improvement decreases as σ increases since the increasing values of σ imply that a larger number of nodes are becoming samplers. Second, the overhead of clustering as well as schedule and model derivation can be observed by comparing the no-sampling and local approaches. Note that the gap between the two is very small and implies that these steps incur very small messaging overhead. Third, the improvement provided by the hybrid approach can be observed by comparing the hybrid and central approaches. We see an improvement ranging from 50 percent to 12 percent to around 0 percent, whereas 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 the messaging cost, especially when σ is small.

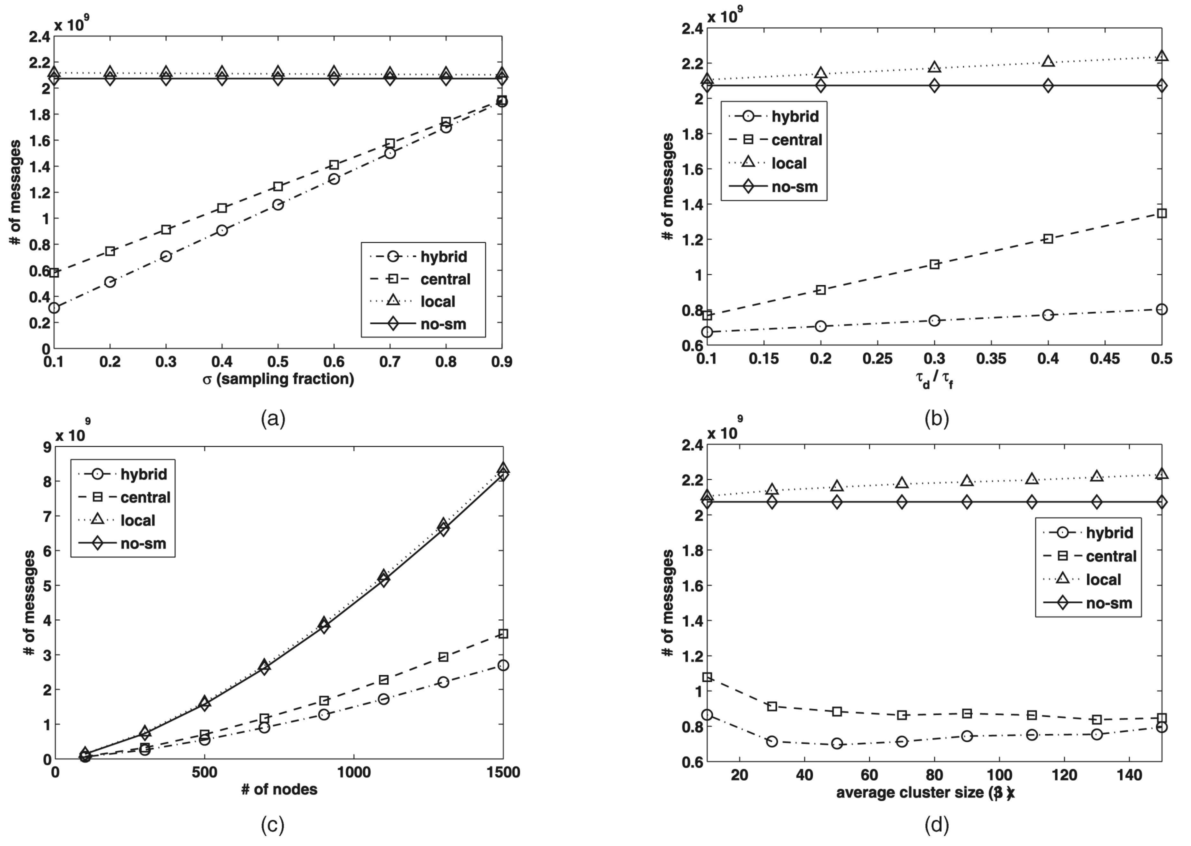

Fig. 3. Messaging cost as a function of the (a) sampling fraction, (b) desired sampling period to forced sampling period ratio, (c) number of nodes, and (d) average subcluster size.

7.1.2 *Effect of the Forced Sampling Period τ_f*

Fig. 3b plots the total number of messages as a function of the desired sampling period to the 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 the local and no-sampling approaches. This is mainly due to the increasing number of forced samples, which results in a higher number of values from the 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 the central approach, ranging from 10 percent to 42 percent, whereas 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.

7.1.3 *Effect of the Total Number of Nodes*

Fig. 3c plots the total number of messages as a function of the total number of nodes. The main observation from the figure is that the central and hybrid approaches scale better with the increasing number of nodes, where the hybrid approach keeps its relative advantage over the central approach (around 25 percent in this case) for different network sizes.

7.1.4 *Effect of the Average Cluster Size*

Fig. 3d plots the total number of messages as a function of the average cluster size (that is, $1/f_c$). 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 the local and no-sampling approaches, we can see a clear overhead that increases with the average 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 around 50 for this scenario, where smaller and larger values increase the messaging cost. It is interesting to note that, in the extreme case, where there is a single cluster in the network, the central and hybrid approaches should converge. This is observed from the right end of the x-axis.

7.2 Results on Energy Consumption

We now present results on energy consumption. We used the ns2 network simulator [22] to simulate the messaging behavior of the ASAP system. The default ASAP parameters were set in accordance with the results from Section 7.1. The radio energy model parameters are taken from [21] (0.0135 Watts for rxPower and 0.02475 Watts for txPower). Moreover, the idle power of the radio in the listening mode is set to be equal to the rxPower in the receive mode. We used the energy-efficient S-MAC [21] protocol at the MAC layer with a duty cycle setting of 0.3. $N = 150$ nodes were placed in a 50 m × 50 m area, forming a uniform grid. The

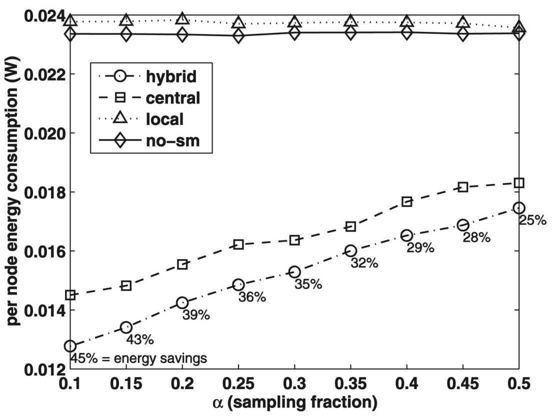

Fig. 4. Energy consumption as a function of sampling fraction.

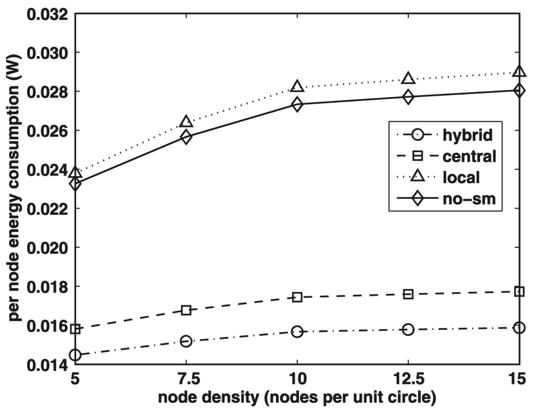

Fig. 5. Energy consumption as a function of node density.

default communication range was taken as 5 m. We study the impact of both the sampling fraction and the node density (by changing the communication range) on the energy consumption of data collection.

7.2.1 Effect of the Sampling Fraction σ

Fig. 4 plots the average per node energy consumption (in watts) as a function of the sampling fraction (σ) for alternative approaches. The results show that ASAP (the hybrid approach) provides between 25 percent to 45 percent savings in energy consumption compared to the no-sampling approach when σ ranges from 0.5 to 0.1. Compare this to the 85 percent to 50 percent savings in the number of messages in Fig. 3. The difference is due to the cost of idle listening. However, given that we have assumed that the listening power of the radio is the same with the receiving power, there is still a good improvement provided by the adaptive sampling framework, which can further be improved with more elaborate power management schemes or more advanced radio hardware.

7.2.2 Effect of Node Density

Fig. 5 plots the average per node energy consumption (in watts) as a function of the node density for alternative approaches. Node density is defined as the number of nodes in a unit circle, which is the circle formed around the node that defines the communication range. The node density is altered by keeping the number of nodes the same but increasing the communication range without altering the power consumption parameters of the radio. We observe in Fig. 5 that the improvement provided by ASAP in terms of the per-node average energy consumption, as compared to that of the no-sampling approach, slightly increases from 38 percent to 44 percent when the node density increases from 5 to 15 nodes per unit circle. Again, the difference between the local and no-sampling approaches is negligible, which attests to the small overhead of the infrequently performed clustering and subclustering steps of the ASAP approach.

7.3 Data Collection Quality

We study the data collection quality of ASAP through a set of simulation-based experiments by using real data. In particular, we study the effect of the data importance factor α on the quality of clustering, the effect of α, β, and subclustering methodology on the prediction error, the trade-off between the network lifetime (energy saving) and prediction error, and the load balance in the ASAP schedule derivation. For the purpose of the experiments presented in this section, 1,000 sensor nodes are placed in a square grid with a side length of 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. The settings of other relevant system parameters are given as follows: TTL is set to 5. The sampling fraction σ is set to 0.5. The subcluster granularity parameter 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 Global Precipitation Climatology Project One-Degree Daily Precipitation Data Set (GPCP 1DD data set) [23]. It provides daily global 1×1 degree gridded fields of precipitation measurements for the three-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 data set has high spatial resolution).

7.3.1 Effect of α on the Quality of Clustering

Fig. 6 plots the average coefficient of variance (CoV) of sensor readings within the same clusters (with a solid line using the left y-axis) for different α values. For each clustering, we calculate the average, maximum, and

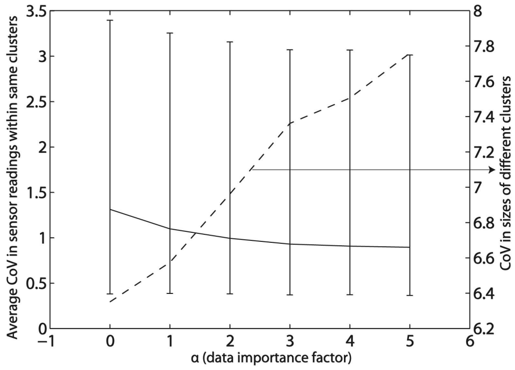

Fig. 6. Clustering quality with varying α.

TABLE 5
Error for Different α Values

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]

minimum of the CoV values of the clusters, where the CoV of a cluster is calculated over the mean data values of the sensor nodes within the cluster. Averages from several clusterings are plotted as an error bar graph in Fig. 6, where the two ends of the error bars correspond to the 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 percent, where a further increase in α does not provide improvement for this experimental setup. To show the interplay between the shape of the clusters and the sensing-driven clustering, Fig. 6 also plots the CoV in the sizes of the clusters (with a dashed line using the right y-axis). With the hop-based clustering (that is, $\alpha = 0$), the cluster sizes are expected to be more evenly distributed, as compared to the sensing-driven clustering. Consequently, the CoV in the sizes of the clusters increases with increasing α, implying that the shape of the clusters are being influenced by the similarity of the sensor readings. These results are in line with our visual inspections in Fig. 2.

7.3.2 Effect of α on the Prediction Error

In order to observe the impact of datacentric clustering on the prediction quality, we study the effect of increasing the data importance factor α on the prediction error. The second column of Table 5 lists the mean absolute deviation (MAD) of the error in the predicted sensor values for different α values listed in the first column. The value of MAD that is relative to the mean of the data values (2.1240) is also given in parentheses in the first column. Although we observe a small improvement of around 1 percent 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 percent confidence interval of the absolute deviation, which is given in the third column of Table 5. The improvement is around 0.87, which corresponds to an improvement of around 25 percent, which is relative to the data mean.

7.3.3 Effect of β on the Prediction Error

As mentioned in Section 4.4, decreasing the subcluster granularity parameter β is expected to increase the effective σ. Higher effective σ values imply a larger number of sampler nodes and thus improves the error in prediction. Fig. 7 illustrates this inference concretely, where the MAD in the predicted sensor 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 the effective σ is plotted

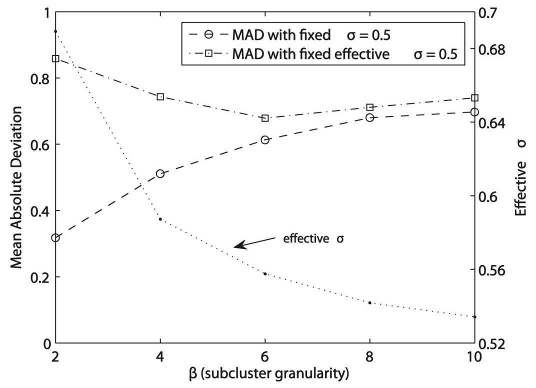

Fig. 7. Effect of β on prediction error.

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 percent (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 percent (0.54 to 0.68). To understand the impact of β better and to decouple it from the number of sampler nodes, we fix the effective σ to 0.5. Fig. 7 plots MAD as a function of β for the fixed effective σ by using a dash-dot line. It is observed that both small and large β values result in a higher MAD, whereas moderate values for β achieve a smaller MAD. This is very intuitive, since small-sized models (small β) are unable to fully exploit the available correlations between the sensor node readings, whereas large-sized models (large β) become ineffective due to the decreased amount of correlation among the readings of large and diverse node groups.

7.3.4 Effect of Subclustering on the Prediction Error

This experiment shows how different subclustering methods can affect the prediction error. We consider three different methods: correlation-based subclustering (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 is a strawman approach that uses purely random assignment to form subclusters. Note that the randomization of the subclustering process will result in sampler nodes being selected almost randomly (excluding the effect of the first-level clustering and node energy levels). Fig. 8 plots MAD as a function of σ for these three different methods of

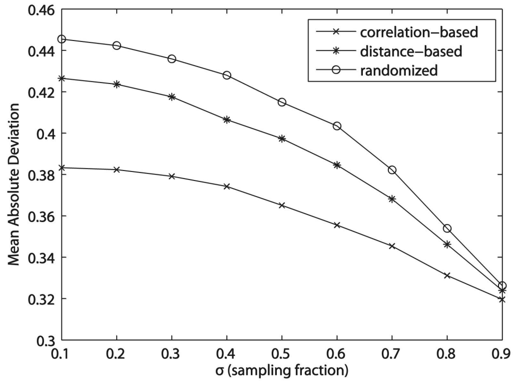

Fig. 8. MAD with different subclusterings.

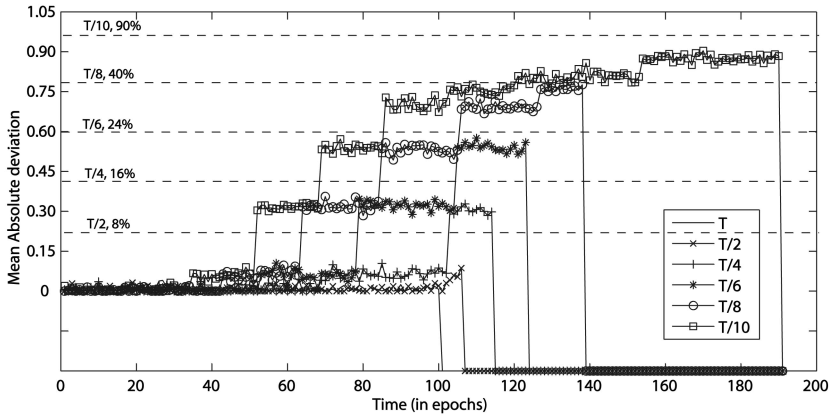

Fig. 9. Prediction error versus lifetime trade-off.

subclustering. The results listed in Fig. 8 are the averages of a large number of subclusterings. We observe that the randomized and distance-based subclusterings perform up to 15 percent and 10 percent worse, respectively, when compared to the correlation-based subclustering in terms of the MAD of the error in value prediction. The differences between these three methods in terms of MAD is largest when σ is smallest and disappears as σ approaches to 1. This is quite intuitive, since smaller σ values imply that the prediction is performed with a smaller number of sampler node values and thus gives poorer results when the subclusterings are not intelligently constructed using the correlations to result in better prediction.

7.3.5 Prediction Error/Lifetime Trade-Off

We study the trade-off between the prediction error and the network lifetime by simulating ASAP 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 ASAP, the average lifetime of the network is $T = 100$ units. This means that the network enables us to collect data with a 100 percent accuracy for 100 time units and then dies out. For comparison, we use ASAP 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. Fig. 9 plots the 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 (as easily read in 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 percent) when ϵ is around 0.4, whereas $T/8$ provides the best improvement (around 40 percent) when ϵ is around 0.8.

7.3.6 Load Balance in Sampler Selection

Although saving battery life (energy) and increasing the average lifetime of the network through the use of ASAP 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 ASAP in achieving the goal of load balance, we compare the variation in the amount of time that the 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 (that is, the percentage of decrease in variance when using our approach compared to the randomized approach) is plotted as a function of β for different σ values in Fig. 10. For all settings, we observe an improvement above 50 percent provided by our sampler selection scheme.

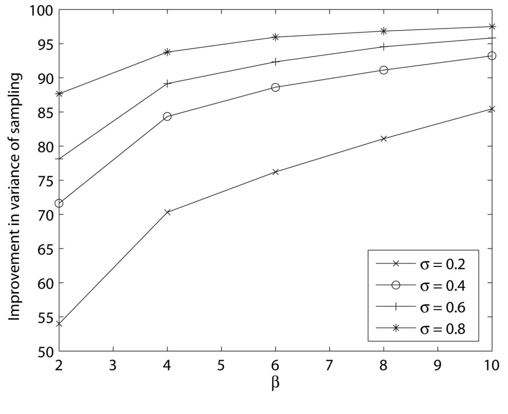

Fig. 10. Load balance in schedule derivation.

8 Related Work

In Section 1.1, we have discussed the distinction between ASAP and other types of data collection approaches such as those based on event detection [24] and in-network aggregation [11]. In summary, ASAP is designed for an energy-efficient periodic collection of raw sensor readings from the network for the purpose of performing a detailed data analysis that cannot be done using in-network executed queries or locally detected events. The energy saving is a result of trading off some level of data accuracy in return for an increased network lifetime, which is achieved by using a dynamically changing subset of the nodes as samplers. This is in some ways similar to previously proposed energy-saving sensor network topology formation algorithms such as probing environment and adaptive sleeping (PEAS) [25], where only a subset of the nodes are made active while preserving the network connectivity. ASAP uses a similar logic but in a different context and for a different purpose: Only a subset of the nodes are used to actively sense, whereas the quality of the collected data is kept high by using locally constructed probabilistic models to predict the values of the nonsampler nodes.

In Section 1.2, we have discussed the relation of ASAP to other model-based data collection frameworks such as BBQ [4] and Ken [13]. In summary, our solution is based on the internode modeling approach like [13], where models are built to capture and exploit the spatial and temporal correlations among the same type of sensor readings of different nodes. This is unlike BBQ [4], where multisensor nodes are assumed, and intranode correlations among the readings of different type sensors within the same node are modeled. Our approach differs from Ken [13] with respect to where in the system the probabilistic models are constructed. Ken builds probabilistic models centrally and does not revise these models under changing system dynamics. We show that, for deployments with high system dynamics, a localized approach like ours can perform adaptation with small overhead thanks to our novel datacentric cluster construction and correlation-based subclustering algorithms.

There are a number of other recent works [26], [27] that have considered the trade-off between energy consumption and data collection quality. In [26], algorithms are proposed to minimize the sensor node energy consumption in 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. ASAP is not bound to queries and collects data periodically so that both online and offline applications can make use of the collected data.

Snapshot Queries [27] is another work that is relevant to ours. In [27], each sensor node is either represented by one of its neighbors or it is a representative node. Although this division is similar to the sampler and nonsampler nodes in ASAP, there is a fundamental difference. The neighboring relationship imposed on representative nodes implies 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 ASAP since the selection process is supported by a clustering algorithm and is not limited to one-hop neighborhoods. In [27], 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 [11] or require the collection of readings from all nodes. Thus, ASAP employs a hybrid approach where prediction is performed outside the network. Moreover, the model-based prediction performed in ASAP uses the correlation-based schedule derivation to subcluster nodes into groups based on how good these nodes are in predicting each other's value. As opposed to this, Snapshot Queries does not use a model and instead employs binary linear regression for each representative-dependent node pair.

Our adaptive 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 nonsampler sensor nodes. Several recent works have also proposed the use of probabilistic inference techniques to learn unknown variables within sensor networks [28], [29], [30], [31]. In [29], 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 [31], probabilistic models representing the correlations between the sensor readings at various locations are used to perform distributed calibration. In [30], 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 [28], 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.

The literature includes many works on clustering, sampling, and prediction. However, the novelty of our approach is in applying these concepts in the context of sensor networks and showing that they can be performed in an in-network manner without centralized control. For instance, the first layer of clustering presented as part of ASAP helps reduce the global problem into a set of localized problems that can be solved in a decentralized way by using the cluster heads. The model generation and the sampler selection are performed by the cluster heads and require communication with only the nodes within a cluster. Our experimental results show that the overall approach significantly benefits the messaging cost of the data collection applications.

9 Conclusion

We introduced an *adaptive sampling* approach for energy-efficient periodic data collection in sensor networks, called ASAP. We showed that ASAP can be effectively used to increase the network lifetime while still keeping the quality of the collected data high. We described three main mechanisms that form the crux of ASAP. 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 the MVN models that capture the correlations among the sensor readings within same subclusters. Last, *adaptive data collection and model-based prediction* is used to minimize the number of messages used to collect data from the network, where the values of the nonsampler nodes are predicted at the base node by using the MVN models. Different from any of the previously proposed approaches, ASAP can revise the probabilistic models through the use of in-network algorithms with low messaging overhead compared to centralized alternatives.

Acknowledgments

This work is partially sponsored by grants from the US National Science Foundation Computer Systems Research (NSF CSR), NSF Information Technology Research (ITR), NSF Cyber Trust, NSF Small Grant for Exploratory Research (SGER), IBM Shared University Research (SUR), and US Air Force Office of Scientific Research (AFOSR).

References

[1] D. Estrin, D. Culler, K. Pister, and G. Sukhatme, "Connecting the Physical World with Pervasive Networks," *IEEE Pervasive Computing,* vol. 1, no. 1, Jan. 2002.

[2] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," *Proc. First ACM Workshop Wireless Sensor Networks and Applications (WSNA '02),* 2002.

[3] M. Batalin, M. Rahimi, Y. Yu, S. Liu, G. Sukhatme, and W. Kaiser, "Call and Response: Experiments in Sampling the Environment," *Proc. Second ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys '04),* 2004.

[4] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model-Driven Data Acquisition in Sensor Networks," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB '04),* 2004.

[5] Taos Inc., *Ambient Light Sensor (ALS),* http://www.taosinc.com/images/product/document/tsl2550-e58.pdf, Dec. 2004.

[6] D. Estrin, R. Govindan, J.S. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," *Proc. ACM MobiCom,* 1999.

[7] D.J. Abadi, S. Madden, and W. Lindner, "REED: Robust, Efficient Filtering and Event Detection in Sensor Networks," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB '05),* 2005.

[8] D. Li, K. Wong, and Y. Hu, A. Sayeed, "Detection, Classification, Tracking of Targets in Micro-Sensor Networks," *IEEE Signal Processing Magazine,* Mar. 2002.

[9] J. Liu, J. Reich, P. Cheung, and F. Zhao, "Distributed Group Management for Track Initiation and Maintenance in Target Localization Applications," *Proc. Second IEEE Int'l Workshop Information Processing in Sensor Networks (IPSN '03),* 2003.

[10] L. Liu, C. Pu, and W. Tang, "Continual Queries for Internet Scale Event-Driven Information Delivery," *IEEE Trans. Knowledge and Data Eng.,* vol. 11, no. 4, July-Aug. 1999.

[11] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "Tag: A Tiny Aggregation Service for Ad Hoc Sensor Networks," *Proc. Fifth Usenix Symp. Operating Systems Design and Implementation (OSDI '02),* 2002.

[12] S. Madden, R. Szewczyk, M. Franklin, and D. Culler, "Supporting Aggregate Queries over Ad Hoc Wireless Sensor Networks," *Proc. Fourth IEEE Workshop Mobile Computing Systems and Applications (WMCSA '02),* 2002.

[13] D. Chu, A. Deshpande, J.M. Hellerstein, and W. Hong, "Approximate Data Collection in Sensor Networks Using Probabilistic Models," *Proc. 22nd IEEE Int'l Conf. Data Eng. (ICDE '06),* 2006.

[14] A. Deshpande, C. Guestrin, and S.R. Madden, "Using Probabilistic Models for Data Management in Acquisitional Environments," *Proc. Second Biennial Conf. Innovative Data Systems Research (CIDR '05),* 2005.

[15] A. Deshpande, C. Guestrin, W. Hong, and S. Madden, "Exploiting Correlated Attributes in Acquisitional Query Processing," *Proc. 21st IEEE Int'l Conf. Data Eng. (ICDE '05),* 2005.

[16] T. Arici, B. Gedik, Y. Altunbasak, and L. Liu, "PINCO: A Pipelined In-Network Compression Scheme for Data Collection in Wireless Sensor Networks," *Proc. 12th IEEE Int'l Conf. Computer Comm. and Networks (ICCCN '03),* 2003.

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

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

[19] Moteiv Corp., *Telos B Datasheet,* http://www.moteiv.com/pr/2004-12-09-telosb.php, Dec. 2004.

[20] O. Chipara, C. Lu, and G.-C. Roman, "Efficient Power Management Based on Application Timing Semantics for Wireless Sensor Networks," *Proc. 25th IEEE Int'l Conf. Distributed Computing Systems (ICDCS '05),* 2005.

[21] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proc. IEEE INFOCOM,* 2002.

[22] *The Network Simulator—ns-2,* http://www.isi.edu/nsnam/ns/, Jan. 2006.

[23] *Global Precipitation Climatology Project,* http://www.ncdc.noaa.gov/oa/wmo/wdcamet-ncdc.html, Dec. 2004.

[24] C.I. Ramesh Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. ACM MobiCom,* 2000.

[25] F. Ye, G. Zhong, S. Lu, and L. Zhang, "Peas: A Robust Energy Conserving Protocol for Long-Lived Sensor Networks," *Proc. 23rd IEEE Int'l Conf. Distributed Computing Systems (ICDCS '03),* 2003.

[26] Q. Han, S. Mehrotra, and N. Venkatasubramanian, "Energy Efficient Data Collection in Distributed Sensor Environments," *Proc. 24th IEEE Int'l Conf. Distributed Computing Systems (ICDCS '04),* 2004.

[27] Y. Kotidis, "Snapshot Queries: Towards Data-Centric Sensor Networks," *Proc. 21st IEEE Int'l Conf. Data Eng. (ICDE '05),* 2005.

[28] M. Paskin and C. Guestrin, "A Robust Architecture for Distributed Inference in Sensor Networks," *Proc. Fourth IEEE Int'l Workshop Information Processing in Sensor Networks (IPSN '05),* 2005.

[29] C. Guestrin, R. Thibaux, P. Bodik, M.A. Paskin, and S. Madden, "Distributed Regression: An Efficient Framework for Modeling Sensor Network Data," *Proc. Third IEEE Int'l Workshop Information Processing in Sensor Networks (IPSN '04),* 2004.

[30] L. Xiao, S. Boyd, and S. Lall, "A Scheme for Robust Distributed Sensor Fusion Based on Average Consensus," *Proc. Fourth IEEE Int'l Workshop Information Processing in Sensor Networks (IPSN '05),* 2005.

[31] V. Byckovskiy, S. Megerian, D. Estrin, and M. Potkonjak, "A Collaborative Approach to In-Place Sensor Calibration," *Proc. Second IEEE Int'l Workshop Information Processing in Sensor Networks (IPSN '03),* 2003.

Buğra Gedik received the BS degree in computer science from Bilkent University, Ankara, Turkey, and the PhD degree in computer science from the College of Computing, Georgia Institute of Technology, Atlanta. He has served on the program committees of several international conferences, such as the IEEE International Conference on Data Engineering (ICDE), the International Conference on Mobile Data Management (MDM), and the International Conference on Collaborative Computing (CollaborateCom). He was a cochair of the First International Workshop on Scalable Stream Processing Systems (SSPS 2007) and on the program committee of the First International Workshop on Distributed Event Processing, Systems and Applications (DEPSA 2007). He is currently a member of the Software Tools and Techniques Group, IBM Thomas J. Watson Research Center. His research interests are in data-intensive distributed computing systems, spanning datacentric overlay networks, mobile and sensor-based data management, and data stream processing, particularly in developing system-level architectures and techniques to address scalability problems in distributed continual query systems and information monitoring applications. He received the Best Paper Award at the 23rd International Conference on Distributed Computing Systems (ICDCS 2003). He is a member of the IEEE.

Ling Liu is an associate professor in the College of Computing, Georgia Institute of Technology, Atlanta, where she directs research programs in the Distributed Data Intensive Systems Laboratory (DiSL), examining performance, security, privacy, and data management issues in building large-scale distributed computing systems. She and the DiSL research group have been working on various aspects of distributed data intensive systems, ranging from decentralized overlay networks, mobile computing and location-based services, sensor network, and event stream processing to service-oriented computing and architectures. Her research group has produced a number of open source software systems, the most popular of which includes WebCQ and XWRAP Elite. She has played key leadership roles on the program, steering, and organizing committees of several IEEE conferences, including the IEEE International Conference on Data Engineering (ICDE), the IEEE International Conference on Distributed Computing (ICDCS), the International Conference on Web Services (ICWS), and the International Conference on Collaborative Computing (CollaborateCom). She is currently on the editorial board of several international journals, including the *IEEE Transactions on Knowledge and Data Engineering* and the *International Journal of Very Large Database Systems*. She has published more than 200 international journal articles and conference proceedings in Internet computing systems, Internet data management, distributed systems, and information security. She has received distinguished service awards from both the IEEE and the ACM. She was the recipient of the Best Paper Award at the 13th IEEE International World Wide Web Conference (WWW 2004), the Best Paper Award at ICDCS 2003, the Pat Goldberg Memorial Best Paper Award in 2005, and the IBM Faculty Award in 2003 and 2006. Her research is primarily sponsored by the US National Science Foundation, DARPA, the US Department of Energy (DoE), and IBM. She is a senior member of the IEEE.

Philip S. Yu received the BS degree in electrical engineering from National Taiwan University, Taipei, the MS and PhD degrees in electrical engineering from Stanford University, California, and the MBA degree from New York University. He is currently the manager of the Software Tools and Techniques Group, IBM Thomas J. Watson Research Center. He is an associate editor of the *ACM Transactions on the Internet Technology* and the *ACM Transactions on Knowledge Discovery from Data*. He is on the steering committee of the IEEE Conference on Data Mining and was a member of the IEEE Data Engineering Steering Committee. He was the editor-in-chief of the *IEEE Transactions on Knowledge and Data Engineering* from 2001 to 2004, and was an editor, an advisory board member, and also a guest coeditor of the special issue on mining of databases. He also served as an associate editor of *Knowledge and Information Systems*. In addition to serving as a program committee member for various conferences, he was the program chair or cochair of the First IEEE International Workshop on Scalable Stream Processing Systems (SSPS '07), the 2006 IEEE International Workshop on Mining Evolving and Streaming Data (IWMESD), the 2006 IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services (CEC/EEE 2006), the 11th IEEE International Conference on Data Engineering (ICDE 1995), the Sixth Pacific Area Conference on Knowledge Discovery and Data Mining (PAKDD 2004), the Ninth ACM Sigmod Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD 2004), the Second IEEE International Workshop on Research Issues on Data Engineering: Transaction and Query Processing (RIDE-TQP 1992), the PAKDD Workshop on Knowledge Discovery from Advanced Databases, and the Second IEEE International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems (WECWIS 2000). He served as the general chair or cochair of the 15th ACM Conference on Information and Knowledge Management (CIKM 2006), ICDE 1998, and the Second IEEE International Conference on Data Mining (ICDM 2002). His research interests include data mining, Internet applications and technologies, database systems, multimedia systems, parallel and distributed processing, and performance modeling. He has published more than 500 papers in refereed journals and conference proceedings. He is the holder of more than 300 US patents. He is a fellow of the ACM and the IEEE. He has received several IBM honors, including two IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, two Research Division Awards, and the 88th Plateau of Invention Achievement Award. He received the Research Contributions Award at ICDM in 2003 and an IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts" in 1999. He is an IBM Master Inventor.