The Clustered AGgregation (CAG) Technique Leveraging Spatial and Temporal Correlations in Wireless Sensor Networks

SUNHEE YOON and CYRUS SHAHABI
University of Southern California

Sensed data in Wireless Sensor Networks (WSN) reflect the spatial and temporal correlations of physical attributes existing intrinsically in the environment. In this article, we present the Clustered AGgregation (CAG) algorithm that forms clusters of nodes sensing similar values within a given threshold (spatial correlation), and these clusters remain unchanged as long as the sensor values stay within a threshold over time (temporal correlation). With CAG, only one sensor reading per cluster is transmitted whereas with Tiny AGgregation (TAG) all the nodes in the network transmit the sensor readings. Thus, CAG provides energy efficient and approximate aggregation results with small and often negligible and bounded error.

In this article we extend our initial work in CAG in five directions: First, we investigate the effectiveness of CAG that exploits the temporal as well as spatial correlations using both the measured and modeled data. Second, we design CAG for two modes of operation (interactive and streaming) to enable CAG to be used in different environments and for different purposes. Interactive mode provides mechanisms for one-shot queries, whereas the streaming mode provides those for continuous queries. Third, we propose a fixed range clustering method, which makes the performance of our system independent of the magnitude of the sensor readings and the network topology. Fourth, using mica2 motes, we perform a large-scale measurement of real environmental data (temperature and light, both indoor and outdoor) and the wireless radio reliability, which were used for both analytical modeling and simulation experiments. Fifth, we model the spatially correlated data using the properties of our real world measurements.

Our experimental results show that when we compute the average of sensor readings in the network using the CAG interactive mode with the user-provided error threshold of 20%, we can save 68.25% of transmissions over TAG with only 2.46% inaccuracy in the result. The streaming mode of CAG can save even more transmissions (up to 70.24% in our experiments) over TAG, when data shows high spatial and temporal correlations. We expect these results to hold in reality, because we used the mica2 radio profile and empirical datasets for our simulation study. CAG is the first system that leverages spatial and temporal correlations to improve energy efficiency of in-network aggregation. This study analytically and empirically validates CAG's effectiveness.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed networks*; C.2.4 [**Computer-Communication Networks**]:

This is an extended version of a paper that appeared in the IEEE International Conference on Communications (ICC), May 2005.
Authors' address: email: sunheeyo@usc.edu.

Distributed Systems—*Distributed databases*; *Distributed applications*; I.6 [**Simulation and Modeling**]

General Terms: Algorithm, Design, Measurement, Performance

Additional Key Words and Phrases: In-network processing and aggregation, clustering, spatial and temporal correlations, energy efficiency, accuracy, approximation, modeling

ACM Reference Format:

Yoon, S. and Shahabi, C. 2007. The Clustered Aggregation (CAG) Technique Leveraging Spatial and Temporal Correlations in Wireless Sensor Networks, ACM Trans. Sens. Netw. 3, 1, Article 3 (March 2007), 39 pages. DOI = 10.1145/1210669.1210672 http://doi.acm.org/ 10.1145/ 1210669.1210672

1. INTRODUCTION

In-network query processing and data aggregation are widely used to save energy, increase scalability, and reduce computation in many monitoring applications of WSN, such as wildlife habitat monitoring [Szewczyk et al. 2004], structural health monitoring [Xu et al. 2004], moving target tracking [Fang et al. 2003], toxic waste monitoring, and seismic monitoring [Husker et al. 2003].

There have been a number of research studies pursuing efficient in-network aggregation in the literature [Intanagonwiwat et al. 2000; Madden et al. 2002; Yao and Gehrke 2003]. TAG, the landmark in-network query processing system, constructs a query routing tree and performs in-network aggregation along the tree. TAG has a fixed set of query operators and a query processor that runs on each node. Alternately, Directed Diffusion allows users to define their own in-network aggregation operators. Neither of these systems exploit the spatial and temporal correlations of data to achieve even more efficient in-network aggregation.

Tobler's first law of geography states that "Everything is related to everything else, but near things are more related than distant things" [Tobler 1970]. This statistical observation implies that data correlation increases with decreasing spatial separation. We proposed a scheme called Clustered AGgregation (CAG) [Yoon and Shahabi 2005b] to improve existing in-network aggregation mechanisms by leveraging this spatial property of sensor data. CAG forms clusters of the sensor nodes sensing similar values and transmits only a single value per cluster as opposed to a single value per node as in TAG-like schemes. Thus, CAG can significantly reduce the number of transmissions, which results in energy savings while incurring a small error in the query result. A *user-provided error threshold*, τ, is a parameter provided by the user to describe the accuracy requirement of the result. This error threshold is used while building clusters such that the difference among the sensor readings in a cluster is bounded by this threshold. This ensures that the resulting approximate answer *always* stays within the error threshold of the correct answer.

In this article, we extend CAG to operate in two modes, interactive and streaming, depending on the dynamics of the environment. In the interactive mode, users issue a one-shot query and the network responds with a single set of responses. This is appropriate for scenarios where the environment (network topology and data) changes dynamically, or users desire to change the

approximation granularity or query attributes interactively. On the other hand, in the streaming mode, the clusterheads transmit a stream of responses for a query that is issued just once. This mode of operation is well-suited for static environments where sensor readings do not change frequently and the query remains valid for a certain period of time. Note that the interactive mode of CAG only exploits the spatial correlation of the sensor data to form clusters, whereas the streaming mode of CAG leverages both temporal and spatial correlations. The latter adjusts clusters locally as the data and topology change over time. The cluster adjustments are infrequent when the data is correlated both spatially and temporally. In the interactive mode, the same weight is assigned to all the clusterhead readings regardless of the cluster size, which can result in large errors while computing an aggregate. In the streaming mode, we count the number of nodes per cluster and use the cluster size as a weight of the clusterhead readings while computing the aggregate function. Therefore, the results of the streaming mode are more accurate than those of the interactive mode.

The advantage of CAG is the high precision of the approximate results. CAG interactive mode guarantees the error in the result is bounded by the user-provided threshold for exemplary and duplicate insensitive aggregation operators such as MAX and MIN. Although the interactive mode of CAG is oblivious to the number of nodes within a cluster, it still provides a good approximation for the summary and duplicate sensitive operators such as SUM, AVG, STD, and VAR when the data is normally distributed and highly correlated. However, the interactive mode cannot guarantee that error in the result is within the error threshold for the summary and duplicate sensitive operators while the streaming mode can. Thus, the errors are bounded while still saving a significant number of message transmissions, hence energy. This benefit amplifies when the number of sensor nodes, the density of node deployment, and the level of data correlation (both spatial and temporal) increase.

CAG is a mechanism to implement within the existing sensor network query system without OS/infrastructure modification. To the best of our knowledge, CAG is a novel system for efficient approximation of in-network aggregation, in that it supports *semantic broadcast* [Woo et al. 2004] by leveraging both the spatial and temporal correlations prevalent in real world data.

This article subsumes our previous work [Yoon and Shahabi 2005b]. The previous version of CAG was not evaluated based on systematic large scale measurement of sensor data. Even though the error in the result for MIN and MAX was bounded by the user-provided error threshold, we discovered that the error for the AVG operator is not bounded by the threshold based on experiments using measured data from Great Duck Island [Mainwaring et al. 2005]. We address this problem by proposing the streaming mode of CAG, which computes the number of nodes within a cluster in order to assign the proper weight to the clusterhead values while computing aggregates, thereby resulting in highly accurate results. In addition, with our previous method of clustering results in variable range clusters, the accuracy of the results depends on the magnitude of the clusterhead sensor value. This made our earlier analysis of CAG intractable for multi-hop topologies. To solve this problem, we propose the fixed range clustering.

We summarize the main contributions of this article as follows:

—Spatio-temporal correlation: We developed the CAG algorithm, which exploits temporal as well as spatial correlations. We investigated the efficiency of CAG using both measured and modeled data.

—Customization: We designed two modes of CAG (interactive and streaming) that are appropriate for different applications and environmental characteristics.

—Clustering independent of data magnitude: CAG groups nodes into the same cluster if their sensor readings fall within a fixed range. Fixed range clustering, as opposed to the variable range clustering from our previous work, ensures that CAG performs well independently of the magnitude of sensor readings and network topology.[1]

—Measurement: We performed a large-scale measurement of the environmental data (temperature and light, both indoor and outdoor) using mica2 motes. We also measured the wireless link reliability using mica2 radios at different transmission powers.

—Model: We derived the data models from the sensor data we obtained from the real-world measurements. Our data model captures two kinds of spatial property: linear and spherical.

Our experimental results for AVG operator with measured data and link reliability, indicate that CAG, with an error threshold of 20%, can save up to 68.25% of transmissions over TAG in the interactive mode with only 2.46% inaccuracy in the result. The streaming mode of CAG can save up to 70.24% over the TAG when data shows high spatial and temporal correlations in just over 100 minutes. CAG used 19.0% less transmission than TAG with the real world dataset from Great Duck Island while incurring less than 6.26% error.

The remainder of this article is organized as follows. Section 2 presents related work and Section 3 describes the two modes of CAG algorithm, interactive and streaming. Section 4 describes the efficiency and accuracy analysis of CAG. Section 5 presents the preliminaries of the spatial correlation models, the data sets and measurements, and our spatial data models. Section 6 describes the evaluation metrics and our experimental setup, and reports the results from our experiments on both interactive and streaming modes. Section 7 concludes this article.

2. RELATED WORK

Figure 1 presents a classification of major query systems in WSN and places our work in the context of related work.

Several in-network aggregation techniques have been proposed for energy-efficient communication in WSN. TinyDB [Madden et al. 2002], Directed-diffusion [Intanagonwiwat et al. 2000], and Cougar [Yao and Gehrke 2003] are the first generation of in-network aggregation systems. These approaches use tree or Directed Acyclic Graph (DAG) topology as an underlying routing

[1]Sections 4 and 6.1 describe this property in detail.

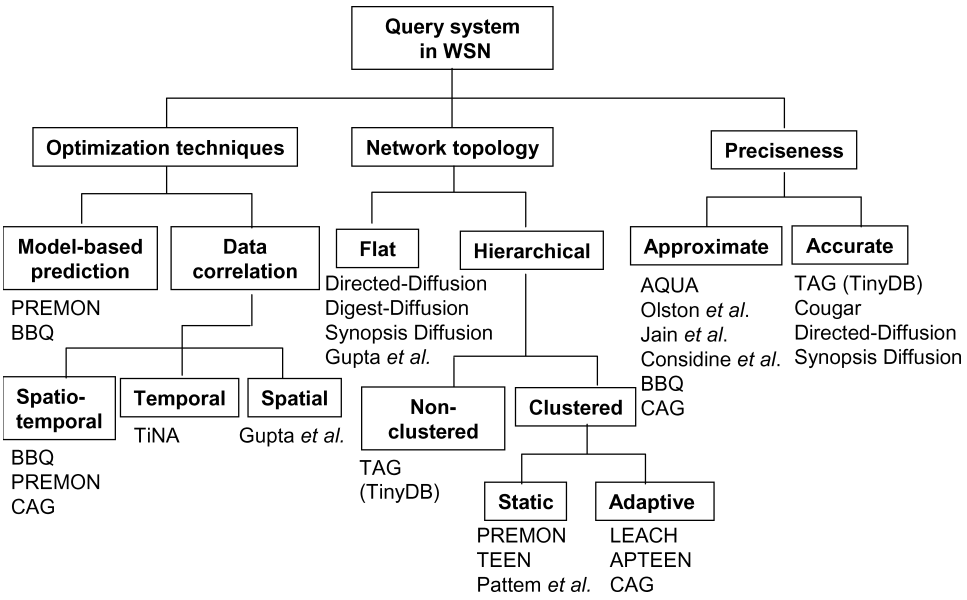


Fig. 1. Classification of major query systems in WSN.

framework. However, they do not consider further energy optimization using spatial or temporal correlation and approximate aggregation.

Two recent approaches, Digest Diffusion [Zhao et al. 2003] and Synopsis Diffusion [Nath et al. 2004], support robust communication for duplicate-insensitive and duplicate-sensitive aggregates, respectively. However, these two systems are not without energy overhead. Digest Diffusion requires each node to maintain link quality statistics to construct the routing tree. Synopsis Diffusion with adaptive ring topology uses redundant transmissions and receptions, which makes it less energy efficient than TAG. Our work shows that CAG is more energy efficient than TAG. Thus, we establish that CAG is more efficient than Synopsis Diffusion.

There has been a variety of research on optimizing tree-based routing. Cristescu et al. [2004] proposed a distributed approximation algorithm for correlated data-gathering in a tree topology. The authors suggested a coding strategy based on a Slepian-Wolf model and a joint entropy coding model, to jointly optimize the transmission structure of the tree and data allocation at a node. Goel and Estrin [2003] proposed a randomized tree structure algorithm that simultaneously optimizes all concave nondecreasing aggregate functions. CAG's approach is different: it is not about optimizing the routing tree by forming better trees; it is about optimizing (minimizing) both energy usage (forwarding) and resulting error using an existing query routing tree.

Allowing for an approximate result instead of requiring an exact answer enables designing energy-efficient in-network aggregation mechanisms. Approximate results can be used in an interactive setting in which users may first ask for a rough picture of regional data before they decide to drill-down

further [Ganesan et al. 2003]. In this scenario, not all sensed data is required to compute the synopsis. Considine et al. [2004] proposed an approximate aggregation technique by generalizing the Flajolet and Martin duplicate-insensitive sketches for duplicate-sensitive aggregates. The AQUA project [Gibbons 2001] proposed a technique to compute a synopsis using a subset of nodes in the network. This technique provides accurate approximate answers to the query. However, it does not leverage any data correlation. It is known that both energy efficiency and accuracy are important in time-critical monitoring. In many systems, however, higher accuracy comes at a prohibitive energy cost. CAG addresses this problem by providing bounded approximate results with significant energy reduction.

Olston et al. [2003] designed an adaptive bounded-width filter in which filter widths are continuously adjusted to match the current data dynamics. In this scheme, distributed data streams transmit a bounded approximate answer to the centralized site with reduced overhead. Jain et al. [2004] tried to minimize resource usage while satisfying the precision requirement by designing a prediction system using Dual Kalman Filter (DKF). A sophisticated filter or prediction scheme can be incorporated in WSN to prevent unnecessary data transmission. The user-provided error threshold τ functions as a fixed-width filter to determine the allowable sensor readings in a CAG cluster.

There have been many studies modeling the spatial correlation property in the context of WSN. Deshpande et al. [2004] proposed a model-based data acquisition prototype called BBQ which uses a *time-varying multivariate Gaussian model*. The authors proposed a framework that can use any data model to predict the sensor readings. Unlike CAG, BBQ does not form clusters; neither does it model the environmental data or take into account the data dynamics. Moreover, BBQ cannot detect outliers whereas CAG can. Guestrin et al. [2004] proposed distributed regression, which is an efficient and general framework for in-network modeling of sensor data. In this work, rather than communicating the data, nodes communicate constraints on the model parameters, thereby significantly reducing the communication cost. Jindal and Psounis [2004] proposed a method to generate spatially correlated data based on a mathematical model. Lennon [2000] describes a technique to generate a synthetic data set that contains realistic spatial patterns with known spatial properties bearing the fractal pattern in the environment. This synthetic pattern is generated using stochastic noise; fractal (strictly quasi fractal) is brown noise in terms of the color of the spectra. Rossi et al. [2004] modeled the diffusion phenomena (such as the propagation of a gas in the air or of a chemical agent in water) using partial differential equations in order to estimate parameters for diffusion monitoring. Fernandez and Green [2002] modeled spatially correlated data in a Bayesian framework using a statistical approach. In this study, we statistically model the spatial correlation property with the measured environmental data using variogram and PDF (Probability Density Function) as a function of internode distance. We observed two correlation data models, linear and spherical, which describe the data from our measurement study.

Techniques such as Low-Energy Adaptive Clustering Hierarchy (LEACH) [Heinzelman et al. 2000], Threshold sensitive Energy Efficient sensor Network protocol (TEEN) [Manjeshwar and Agrawal 2001], and Adaptive TEEN (APTEEN) [Manjeshwar and Agrawal 2002] use hierarchical clusters and routing to save energy. LEACH [Heinzelman et al. 2000] forms clusters based on the received signal strength and uses local clusterheads as routers. Transmissions are made only by clusterheads. LEACH utilizes randomized rotation of local clusterheads to evenly distribute the energy overhead among the sensors in the network. The main difference between LEACH and CAG is that LEACH does not provide a mechanism to compute aggregate using clusterhead values, while CAG does. TEEN [Manjeshwar and Agrawal 2001] is another hierarchical protocol designed to be responsive to sudden changes in the sensor readings. TEEN, which is based on hierarchical clustering, forms clusters using nearby nodes. The nodes transmit sensor readings only when they fall above the specified threshold (hard threshold) and changes by a given amount (soft threshold). While this saves energy, it does not support periodic reports. APTEEN [Manjeshwar and Agrawal 2002] addresses both periodic data collection and prompt reporting of time-critical events. None of these protocols, however, leverages spatial and temporal correlations to improve efficiency.

Pattem et al. [2004] analyzed the total cost for jointly optimizing routing performance and data compression using the joint entropy of sources, leveraging spatial correlation. Authors claimed that there exists a static, near optimal cluster size for ranges of spatial correlation. In contrast, CAG is an adaptive clustering scheme (clusters adjust over time) with lossy aggregation, which provides an approximate result where the error is bounded by the user-provided error threshold in the streaming mode.

PREMON [Goel and Imielinski 2001] provides energy-efficient monitoring based on a clustered architecture. Clusterhead nodes in PREMON use a technique similar to the MPEG compression algorithm, and generate prediction models to predict the spatio-temporal data within a cluster. PREMON saves energy by avoiding the transmissions of all the redundant data that can be successfully predicted by the clusterhead node. PREMON assumes that the clusters are already formed using any existing mechanism while CAG forms clusters using real-time sensor values. PREMON uses lossy compression with approximation where the error is not bounded (they did not mention or show anything on the boundedness of error), whereas CAG uses lossy approximation and guarantees that the error in the result is bounded by the given error threshold. PREMON uses the block-matching algorithm of MPEG to compute the prediction model, whereas we classify the spatial correlation models existing in the measured sensor reading to investigate the impact of different correlation models on the CAG algorithm.

TiNA [Sharaf et al. 2003] exploits temporal correlation in sensor data while CAG takes advantage of both spatial and temporal correlations in sensor data. Gupta et al. [2005] proposed an efficient data-gathering algorithm exploiting the spatial correlation. However, their algorithm is not based on the clustering technique, and the overhead from selecting the connected

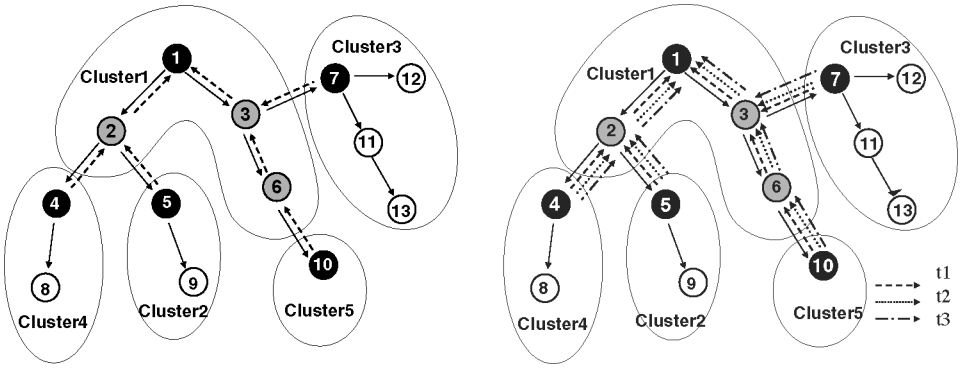


Fig. 2. Two modes of CAG's operation: interactive and streaming modes. The solid lines indicate the query propagation and the dotted lines indicate the response. Black nodes are clusterheads, gray nodes are bridges, and white nodes are nonparticipating nodes.

correlation-dominating set compromises the efficiency of the proposed algorithm. In addition, their work is not validated empirically using the measured sensor data and does not address the data dynamics.

CAG exploits semantic broadcast [Woo et al. 2004] in order to reduce the communication overhead by leveraging spatial and temporal correlations. CAG achieves efficient in-network processing by allowing a unified mechanism between query routing (networking) and query processing (application). Instead of gathering and compressing all the data (lossless algorithm), CAG generates a synopsis by filtering out insignificant elements in data streams (lossy algorithm) to minimize response time, storage, computation, and communication costs.

3. THE CAG ALGORITHM

CAG, originally introduced in Yoon and Shahabi [2005b], is an algorithm to compute approximate answers to queries by using representative values in the network and leveraging spatial and temporal properties of data. The prevalence of spatial and temporal correlations in environmental phenomena makes it possible for CAG to ignore redundant data and quickly generate a synopsis of the data distribution with significant energy savings. We use the AVG as a main aggregate operator to describe, analyze, evaluate CAG, and to compare CAG with other algorithms. Thus, the AVG operator is implied in all discussions unless we explicitly mention other operators.

3.1 Two Modes of CAG Operation

Depending on the application's requirements, CAG can work in two modes: *interactive and streaming*. CAG generates a single set of responses for a query in the interactive mode. In the streaming mode, periodic responses are generated in response to a query. The interactive mode of CAG exploits only the spatial correlation of sensed data, whereas the streaming mode of CAG takes advantage of both spatial and temporal correlations of data. Figure 2(a) shows the clusterhead responding with a single value in the interactive mode.

Figure 2(b) shows periodic response messages from clusterheads in the streaming mode.

The CAG algorithm operates in two phases: query and response. During the query phase, CAG forms clusters when a TAG-like forwarding tree is built using a user-specified error threshold τ. In the response phase, CAG transmits a *single* value per cluster. CAG is a *lossy clustering* method in that only the clusterheads contribute to the aggregation.

3.2 Interactive Mode

A user runs the CAG algorithm by specifying a query using a syntax similar to that of TAG except there is an additional "*threshold* τ" clause. The user query can be described as $UQ = <QueryID,\ O_i, \tau\ >$, where O_i is the monitoring attribute and τ is the *user-provided error threshold*. CAG supports disseminating multiple queries with different *QueryID* and O_i. Subsequently, the base station broadcasts the query packet $Q = <UQ, ParentID, MyID, level, CR>$, where ParentID is the node ID of the parent in the forwarding tree, MyID is the node ID of the transmitting node, and *level* is the depth of the current node in the forwarding tree. Note that CR is included in the query to be compared with each MR when it is received by a node. Clusters are formed when the forwarding tree is built.

Algorithm 1 Pseudocode of the interactive mode of CAG algorithm

```
1: Function Query.Received:
2: if MR ∈ [CR − Range × τ, CR + Range × τ] then
3:    clusterhead = FALSE;
4:    broadcast query Q;
5: else
6:    CR = MR;
7:    clusterhead = TRUE;
8:    broadcast query Q;
9: end if
10: Function Response.Received:
11: enque response to the buff;
12: Function ResponseTimer.Fired:
13: if clusterhead then
14:    forward aggregate(buff, MR);
15: else if size(buff) > 0 then
16:    forward aggregate(buff);
17: end if
```

Upon receiving the query, each node decides to join a cluster based on *Clusterhead sensor Reading (CR)* and *My local sensor Reading (MR)*; if $|\ MR - CR| \leq Range \times \tau$, where $Range = MaxValue - MinValue$ of the entire dataset, then the sensor node joins the cluster. This range can be determined in advance by surveying the dataset using MAX or MIN operation of the CAG algorithm. This $Range$ value is based on the range that is semantically meaningful for users or the range of values that ADC sensor provides. The only requirement is that a consistent scale be used for filtering, thresholding, clustering, and displaying.

We define the interval $[CR - Range \times \tau.\ CR + Range \times \tau]$ as the clustering range. Forming clusters using τ is termed τ-approximation, because τ also functions as the error bound of the result such that $|EstimatedResult - CorrectResult| < \tau$. This is why τ, a user-provided error threshold, is interchangeable with a *user-provided error-tolerance threshold*. Algorithm 1 shows the pseudocode of the interactive mode of the CAG algorithm.

Once all the nodes receive the query packet, the response phase starts. The ResponseTimer works the same way as *epoch* timer in TAG [Madden et al. 2002]. On nodes higher up in the tree, the ResponseTimer fires later than on the nodes lower in the tree. Thus, by the time a parent is ready to aggregate and transmit its result, it would have already received the results from its children. Whenever the ResponseTimer fires, only clusterheads transmit packets with the following tuple: $R = <ParentID, CR>$. If the clusterheads cannot communicate with each other, the intermediate nodes, termed *bridge nodes*, are required to *bridge* the segments of the forwarding tree. *Bridge nodes* do not contribute their sensor readings to the aggregate by default, but they can optionally participate in the aggregation because they transmit the packets anyway. A more detailed example of the CAG algorithm execution is described in Yoon and Shahabi [2005b].

In the interactive mode, CAG builds a forwarding tree when a query is sent out. Just like TAG [Madden et al. 2002], we assume an underlying mechanism to blacklist asymmetric links. Thus, the forwarding path is set along the reverse direction of the query propagation. This newly formed clustered tree can address the dynamics of network and data on the fly. However, the interactive mode requires the overhead for broadcasting a query each time a user wants new data from the network. The frequent rebuilding of the tree can be wasteful if the sensed data is almost the same over time. If the data is unchanged, clusterhead nodes and the forwarding tree are likely to be the same. Moreover, in the interactive mode, CAG does not count the number of nodes within a cluster; this may exchange the accuracy of the results for energy savings by reducing the number of packet transmissions.

In our earlier experiments using the measured data from Great Duck Island [Yoon and Shahabi 2005b], we observed that CAG may result in an out-of-bound error with AVG operation, regardless of the error threshold and data correlation, when the data values do not follow the normal distribution. If the data is normally distributed, it is more likely that the clusterhead values will be closer to the mean than far from it. For a given clusterhead value (which is likely to be close to the mean), more nodes are likely to have their sensor readings close to the clusterhead value compared to the readings derived from uniform distribution. Thus, with normal distribution, the clusterhead value is more representative of all the sensor readings in the cluster, than with uniform distribution. This property results in higher accuracy with normal distribution compared to uniform distribution.

We implemented aggregation operators in CAG as shown in Table I, which is a subset of operators mentioned in Madden et al. [2002]. Duplicate-sensitive operators such as SUM and AVG cannot tolerate duplicate packets, while duplicate insensitive operators can. Exemplary aggregate returns

Table I. Properties of Aggregate Operators Supported by CAG. Taxonomy is Based on Madden et al. [2002]. Key for Abbreviations: Exemplary (E), Summary (S), Distributive (D), and Algebraic (A). The Error for Only the MAX and MIN Operators is Bounded in the Interactive Mode of CAG

Aggregation Operator	MAX	MIN	CNT	SUM	AVG	VAR	STD
Duplicate Sensitive	No	No	Yes	Yes	Yes	Yes	Yes
Exemplary, Summary	E	E	S	S	S	S	S
Monotonic	Yes	Yes	Yes	Yes	No	No	No
Partial State	D	D	D	D	A	A	A
Error Bound	τ	τ	Correct	$N\tau$	τ	$4\tau^2$	2τ

representative sensor readings such as MAX or MIN, while summary aggregate such as AVG and CNT compute the summary of all the sensor readings. Monotonic aggregate monotonically increases or decreases after each intermediate aggregation. The partial state of algebraic operators consists of multiple variables. For example, the partial state of AVG consists of SUM and CNT. The partial state of distributive operators such as CNT constitutes only a single variable.

For MIN operator, each participating node (clusterhead) returns the MIN of all the values received in an epoch. The maximum error bound of MIN returned by CAG is equal to or smaller than τ. MAX operation works in a similar way as MIN, and CAG returns an error equal to or smaller than τ. To compute VAR, CAG adds squared value along the AVG (SUM and CNT) at each node by using the following equation: VAR $= \sum {v_i^j}^2/n$ - AVG^2; and STD is computed as the square root of VAR.

Although CAG interactive mode does not provide the result with the bounded error in duplicate sensitive and summary aggregation operators such as AVG and VAR, it provides the result with bounded error with exemplary and duplicate insensitive operators such as MIN and MAX. Note that the interactive mode only leverages the spatial correlation of data and cannot take advantage of the temporal correlation.

3.3 Streaming Mode

The motivation for the streaming mode is the potential to exploit the temporal correlation of data existing in nature, in addition to the spatial correlation already exploited by the interactive mode of the CAG algorithm. In the streaming mode of CAG, a single query generates periodic responses from the network.

A query for the streaming mode uses the clause "*epoch duration i*" to define the sampling frequency. In response to a query with this clause, the network is expected to generate a query reply every i seconds while the query is injected into the network only once.

The query phase of the CAG algorithm in the streaming mode is identical to that of the clustering algorithm in the interactive mode. The response phase algorithm is the major difference between the streaming mode and the interactive mode. First, streaming mode must task the clusterheads to generate response messages once per epoch, as opposed to the one-shot response in the interactive mode. Second, the clusters need to be updated and repaired, as

sensor readings change over time and become inconsistent for current cluster membership. Third, streaming mode allows amortizing the cost of cluster size estimation over the long expected lifetime of a query. This makes it practical to obtain the cluster size along with clusterhead readings, which enables the streaming mode of CAG to compute results with high accuracy and guarantee that the resulting error is always bounded by the user-given threshold even when both the population and sampled data are not normally distributed.

3.3.1 *Cluster Adjustment.* The purpose of cluster adjustment is to make cluster membership consistent as sensor readings change over time. Note that a sensor reading must satisfy the condition ($|MR - CR| \leq Range \times \tau$) if it is to remain within a cluster. As soon as the reading is outside this range, the node must be evicted from its current cluster. Consequently, the node must be either added to a different cluster where its sensor reading is within the clustering range of the clusterhead value, or start a new cluster with itself as a clusterhead.

Algorithm 2 Pseudocode of the streaming mode of CAG algorithm

```
1: Function Query.Received:
2: if MR ∈ [CR − Range × τ, CR + Range × τ] then
3:    clusterhead = FALSE;
4:    broadcast query Q;
5: else
6:    CR = MR;
7:    clusterhead = TRUE;
8:    broadcast query Q;
9: end if
10: Function Response.Received:
11: update neighbor table;
12: enque response to the buff;
13: Function ClusterAdjustmentMsg.Received:
14: update neighbor table;
15: if (clusterhead_id == ClusterAdjustmentMsg.previous_clusterhead_id)
    then
16:    ClusterAdjustmentMsgReceived = TRUE;
17:    process cluster adjustment;
18: else
19:   ClusterAdjustmentMsgReceived = FALSE;
20: end if
21: Function ClusterAdjustmentTimer.Fired:
22: if MR ∉ [CR − Range × τ, CR + Range × τ] then
23:    if (my reading is within neighbor's clustering range) then
24:       join that neighbor's cluster;
25:       notify cluster membership change to old and new clusterhead;
26:    else
27:       create a new cluster with myself as the clusterhead;
28:       notify cluster membership change to old clusterhead;
29:    end if
30:    if (cluster join or cluster create or ClusterAdjustmentMsgReceived)
       then
21:       propagate ClusterAdjustmentMsg;
32:    end if
33:    end if
```

```
34: Function Epoch.Fired:
35: if clusterhead then
36:    forward aggregate(buff, MR);
37: else if size(buff) > 0 then
38:    forward aggregate(buff);
39: end if
```

The cluster adjustment algorithm works by having the nodes in the network check if their sensor readings are within the allowed clustering range of the clusterhead every *Cluster Adjustment Interval*. If the sensor reading is still within the allowed range, the cluster membership is still valid and no further action is necessary. If the sensor reading has veered off the clustering range, the node first attempts to *migrate* to a neighboring cluster where its sensor reading might be within the range for that cluster. To aid in the discovery of such neighboring clusters and their clustering range, the nodes snoop the broadcast medium while responses are transmitted and clusters are adjusting, and keep track of all the neighboring clusters within its radio range. If a suitable neighboring cluster for its sensor reading is not found, the node must *create* a new cluster with itself as the clusterhead. In either case, the node must inform its children in the routing tree that it is no longer in the old cluster. The node must also inform the children (who in turn will inform their children) the new clusterhead value. If the children find this new clusterhead value compatible with their sensor reading, they will then join the cluster and propagate the message down the tree, otherwise they will proceed to start their own cluster and repeat the algorithm.

There are two properties of the adjustment algorithm that help control the cluster adjustment overhead. First, the cluster adjustment messages need only be propagated to the nodes within a cluster of which the node with the out-of-range reading is a member. Because the change of the clustering range in one cluster does not affect the range in a different cluster, there is no need to propagate these adjustment messages to other clusters. Second, the cluster adjustment timers are orchestrated in such a way that the parent node in the query routing tree always performs cluster adjustment before its children. By the time a node runs the adjustment algorithm, it can be sure that cluster adjustment has already taken place in all the upstream nodes. This ensures that a node needs to perform at most one adjustment every *Cluster Adjustment Interval*. That is, the maximum amount of time that data can be out-of-range of their current cluster depends on the *Cluster Adjustment Interval*. We can select the cluster adjustment interval as a function of the temporal correlation of the data, and the accuracy and agility requirement of the application. In other words, a smaller interval makes the system more responsive to the data dynamics. With these two techniques, adjustment cost can be controlled to make cluster adjustment practical and efficient.

3.3.2 *Cluster Size Estimation.* Errors in the result obtained from the interactive mode can be large because equal weights are assigned to the results

coming from clusters of different sizes. For example, while computing an average, the error can be quite large if we assign equal weight to the result from a large and a small cluster. One way to address this is by estimating the size of the cluster so that an appropriate weight can be assigned to the clusterhead values while computing an aggregate. Unfortunately, cluster size estimation is too costly in the interactive mode as it would require a large number of transmissions to compute the size, which is used a single time in a one-shot query. In streaming mode, the queries last a long period of time. Even though the cost of counting the number of nodes in a cluster for the first time is high, there are two reasons for which cluster size estimation is practical in the streaming mode. First, the amortized cost of cluster size estimation over the duration of the query becomes quite small. Second, it is possible to update the node count incrementally when cluster adjustment algorithm changes the number of nodes in a cluster. With high temporal and spatial correlations, the cluster dynamics are rare, which makes changes in cluster size infrequent and the incremental cluster size estimation overhead small.

When clusters are formed for the first time, each node sends a *count increment* message to its clusterhead. The number of nodes in a cluster is computed by counting the number of *count increment* messages received at the clusterhead. Due to data dynamics, some nodes might leave the cluster and some other nodes might join the cluster. Every time a node joins a different cluster, the node sends a *count decrement* message to its old clusterhead and sends a *count increment* message to the new clusterhead. If a node forms a new cluster with itself as the clusterhead, it sends a *count decrement* message to its old clusterhead. The children in the query routing tree must now use the same algorithm to ensure that the node count remains consistent in the old as well as the new cluster.

Cluster adjustment ensures that the clusterhead value is representative of the readings in a cluster even with changing sensor readings. Cluster size estimation enables CAG to assign appropriate weights to the clusterhead values while computing aggregates. Together these two techniques enable CAG to compute results efficiently with high accuracy and guarantee that the results are within the user-provided thresholds regardless of data distribution.

Table II compares these two modes of CAG operations.

4. ANALYSIS OF CAG: EFFICIENCY AND ACCURACY

In this section, we formally analyze the efficiency and accuracy of the CAG algorithm in terms of the number of transmissions and the absolute error, respectively. We analyze the efficiency of CAG for the interactive mode or a single response of streaming mode.[2] We also prove that the absolute error is always bounded by the given threshold value in the streaming mode even when the data is not normally distributed.

In this section, we define v as the sensor reading at the root node, and v_i^j as the sensor reading of the ith child of v in the jth level in the tree. Even though the

[2]The efficiency of the CAG algorithm for both interactive and streaming modes is investigated in Section 6.2.1 and Section 6.2.2 respectively, using simulations.

Table II. Comparison of Two Modes of CAG's Operation: Interactive and Streaming

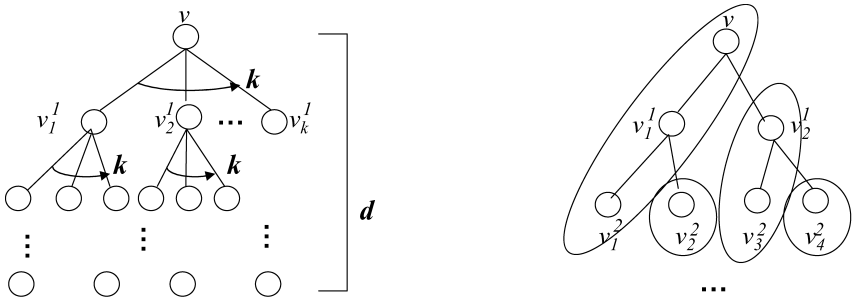
	Interactive Mode	Streaming Mode
Description	Single response for a query	Multiple responses for a query (periodic or event driven)
Exploiting property	Spatial correlation	Spatial and temporal correlations
Clustering	Fixed cluster and clusterhead for a query	Same cluster and clusterhead until reclustering
Advantages	1) Good for reactive, interactive, and one-shot query 2) Good estimation when data is normally distributed with summary and duplicate sensitive operators	1) Good for proactive/reactive (periodic response or event driven) and long-lived query 2) Appropriate for model-based clustering 3) Accurate: bounded error for any data distribution
Disadvantages	1) Not accurate when data is not normally distributed 2) If multiple responses are desired over time, users have to inject queries each time a response is needed	1) Extra overhead and complexity from cluster adjustment and counting the number of nodes in a cluster 2) Unfair energy usage in static environment: Energy bottleneck at clusterhead because clusterhead rarely changes
Operators with bounded error	MAX, MIN	MAX, MIN, CNT, SUM, AVG, VAR, STD

real world is not uniform, in this analysis we assume that the sensor reading v_i^j is independently identically distributed random variable uniformly distributed over the range [0, 1]. The CAG algorithm is designed to take advantage of correlated data. Because there is no spatial correlation between the data with the uniformly distributed independently identically distributed values, the probability of the sensor readings from two nodes are in the same cluster is minimal. Thus, the analysis with uniformly distributed independently identically distributed data will give us an insight into the worst case performance and accuracy of the CAG algorithm.

We assume that each node in the tree has an average branching factor of k. Figure 3(a) depicts a k-ary balanced tree with depth d and Figure 3(b) shows the same tree with annotation using the variables used in this analysis. Let T be the entire tree, and N_T be the number of clusters in T. $N_{v_i^j}$ is the number of clusters in the subtree rooted at node v_i^j, and $n\{v\}$ is the number of nodes that are in the same cluster as v. Even though our analysis does not address the bridge nodes, our simulation does (Figure 11(b)).

To calculate the expected number of transmissions, we need to compute the expected number of clusters. We begin to build clusters from the root node with its single-hop children. As we assumed v has k children on average, N_T is given by:

$$N_T = 1 + N_{v_1^1} + \cdots + N_{v_k^1} - n\{v_i^1 | v_i^1 \text{ is in the same cluster with } v\}. \quad (1)$$



(a) Tree with an average branching factor k and a depth d for the analysis of the number of clusters.

(b) An example of a set of clusters formed in a balanced binary tree.

Fig. 3. Examples of the query routing tree for the analysis of CAG.

When we compute the expected value of Equation (1), we obtain the following equation.

$$E[N_T] = 1 + \sum_{i=1}^{k} E[N_{v_i^1}] - \sum_{i=1}^{k} P_r\big[v_i^1 \text{ is in the same cluster with } v\big]. \quad (2)$$

Here, $P_c = P_r[v_i^j$ is in the same cluster with $v]$; the probability that v_i^j is in the same cluster as v, is derived as follows:

$$\begin{aligned} P_c &= P_r\big[-\tau \le v_i^j - v \le \tau\big] \\ &= \begin{cases} v+\tau, & \text{if } 0 < v < \tau \\ 2\tau, & \text{if } \tau \le v \le 1-\tau \\ 1-v+\tau, & \text{if } 1-\tau < v \le 1 \end{cases} \\ &= \int_0^{\tau} (v+\tau)\,dv + (1-2\tau)2\tau + \int_{1-\tau}^{1} (1-v+\tau)\,dv \\ &= \tau(2-\tau). \end{aligned} \quad (3)$$

Now we analyze the number of transmissions for CAG. As a special case, first we compute the number of clusters only with the single-hop nodes, by extending Equation (2) and combining Equation (3) as follows. In this scenario, all the nodes including the root nodes are within single-hop radio range from any node in the network. Thus, the number of children of the root nodes is $k = |T| - 1$.

$$\begin{aligned} E[N_T] &= 1 + \sum_{i=1}^{k} E[N_{v_i^1}] - \sum_{i=1}^{k} P_c \\ &= 1 + k - k \times P_c \\ &= 1 + (1 - P_c)k \\ &= 1 + (1 - 2\tau + \tau^2)(|T| - 1) \\ &= |T|(1 - 2\tau + \tau^2) + \tau(2-\tau) \\ &= N(1 - 2\tau + \tau^2) + \tau(2-\tau) \\ &= (N-1)\tau(\tau - 1) + N. \end{aligned} \quad (4)$$

We can generalize the single-hop scenario into the multiple-hop scenario by iteratively using Equation (2) for each node in the network up to the depth $d-1$.

$$\begin{aligned} E[N_T] &= 1+\sum_{i=1}^{k} E[N_{v_i^1}] - \sum_{i=1}^{k} P_c \\ &= 1+k-kP_c \quad : \text{for d} = 1 \\ &= 1+(k-kP_c)+k(k-kP_c) \quad : \text{for d} = 2 \\ &= 1+(k-kP_c)+k(k-kP_c)+k^2(k-kP_c) \quad : \text{for d} = 3 \\ &= \textit{etc.} \text{ for d} = 4, 5, \ldots \end{aligned}$$

Thus, $E[N_T]$ for a k-ary balanced tree with depth d is given below:

$$\begin{aligned} E[N_T] &= 1+\frac{(k-kP_c)(k^d-1)}{k-1} \\ &= 1+\frac{k(1-\tau(2-\tau))(k^d-1)}{k-1}. \end{aligned} \tag{5}$$

Equation (5) validates Equation (4), which is for the single-hop scenario as a special case of (5) with $k=|T|-1$ and $d=1$. Thus, Equation (5) estimates the number of clusters in both single-hop and multi-hop topologies.

As in Equation (5), the expected number of clusters depends on the correlation level P_c and the branching factor k where $P_c=\tau(2-\tau)$ and $k<|T|=N$.

The size of a cluster, S_T, can be computed as follows:

$$\begin{aligned} S_T &= \frac{\text{Total number of nodes}}{\text{Expected number of clusters}} \\ &= \frac{N}{E[N_T]} \\ &= \frac{N}{1+\frac{(k-kP_c)(k^d-1)}{k-1}} \\ &= \frac{N(k-1)}{(k-1)+k(1-\tau(2-\tau))(k^d-1)}. \end{aligned} \tag{6}$$

As shown in Equation (6), the size of a cluster, S_T, is also a function of P_c and k where $P_c=\tau(2-\tau)$. With uniformly distributed sensor readings, $P_c=\tau(2-\tau)$ from Equation (3). For empirical data, we can derive P_c as shown in Equations (13) and (14) in section 5.4, and use that P_c to compute S_T.

Because we build clusters based on the absolute range (defined in Table III), the absolute error is always bounded by τ, such that $|v_i^j - v| \leq \tau$ where v_i^j is normalized to [0, 1]. Thus, the sensor reading on each node, v_i^j, can be off by up to τ from the clusterhead reading. If the total number of nodes is N, the maximum error for the entire network becomes $(N-1)\tau$. For AVG operator, the maximum error is $\frac{(N-1)\tau}{N}$. For SUM operator, the maximum error is $(N-1)\tau$. For MIN and MAX, the maximum error is τ:

$$\text{Real MIN} \leq \text{CAG MIN} \leq \text{Real MIN} + \tau$$
$$\text{Real MAX} \geq \text{CAG MAX} \geq \text{Real MAX} - \tau.$$

Table III. The Metrics Used in the Evaluation of CAG

Metrics	Description
Cluster formation range	Absolute range: $CR \pm Range \times \tau$, where $Range = MaxValue - MinValue$
Reduced number of transmissions	$\frac{nTX(TAG)-nTX(CAG)}{nTX(TAG)} \times 100$ in the interactive mode
Number of transmissions	$nTX(CAG)$ in the streaming mode
Accuracy of result	Absolute error: $E_r = \lvert Estimated\,Result - Correct\,Result \rvert \times 100$
Number of bridge nodes	Number of participating nodes - Number of clusterheads
Reduced number of transmissions per density	Three densities: moderate (average 17 neighbors per node), dense (average 26 neighbors per node), and sparse (average 9 neighbors per node).

VAR is given by $\text{VAR} = \sum_1^n (v_i^j - \text{AVG})^2/n$. Here v_i^j can be off from the clusterhead value by up to τ, and the AVG returned by CAG has maximum error of $\frac{(N-1)\tau}{N} < \tau$. Thus, the maximum error for VAR is given by $\sum_1^n (\tau + \tau)^2/n = 4\tau^2$. The maximum error for STD is 2τ. All the operators implemented in CAG and maximum error bounds are summarized in Table I.

Now we can formally prove that the absolute error E_r is always bounded by τ in the streaming mode.[3] Assume that the values in each cluster are sorted. Let v_{ij} be the jth unique value in the cluster i, $n(v_{ij})$ be the number of nodes with jth unique value in the cluster i, and $d(i)$ be the number of unique values in the cluster i. Let C_i be the clusterhead value for the cluster i. Let $c(i)$ be the number of nodes in the cluster i such that $c(i) = \sum_{j=1}^{d(i)} n(v_{ij})$. Let s be the number of clusters in the network, and N be the total number of nodes in the network such that $N = \sum_{i=1}^{s} c(i)$. We can compute the AVG operation correctly using the TAG, and approximately using the CAG as in Equation (7) and (8), respectively.

$$Correct\,Result = \frac{\sum_{i=1}^{s}\sum_{j=1}^{d(i)} n(v_{ij}) \times v_{ij}}{N} \tag{7}$$

$$Estimated\,Result = \frac{\sum_{i=1}^{s} c(i) \times C_i}{N}. \tag{8}$$

The absolute error, E_r, using the CAG can be computed as follows.

$$\begin{aligned} E_r &= \lvert Estimated\,Result - Correct\,Result \rvert \quad \text{, where } 0 < \tau \le 0.5 \\ &= \frac{\left\lvert \sum_{i=1}^{s} c(i)C_i - \sum_{i=1}^{s}\sum_{j=1}^{d(i)} n(v_{ij})v_{ij} \right\rvert}{N} \\ &= \frac{\left\lvert \sum_{i=1}^{s}\sum_{j=1}^{d(i)} n(v_{ij})C_i - \sum_{i=1}^{s}\sum_{j=1}^{d(i)} n(v_{ij})v_{ij} \right\rvert}{N} \\ &= \frac{\sum_{i=1}^{s}\sum_{j=1}^{d(i)} n(v_{ij})\lvert C_i - v_{ij} \rvert}{N} \end{aligned}$$

[3]We mentioned in Section 3.2 that interactive mode may not provide the bounded error when the data is not normally distributed.

$$\leq \frac{(N-1)\tau}{N}$$
$$\leq \tau. \quad (9)$$

As shown in Equation (9), E_r is always bounded by τ when it computes AVG in the streaming mode regardless of the data distribution.

Note that the actual magnitude of absolute error with the measured sensor data and without packet loss in the streaming mode is a constant for the linear model as in Equation (15) and a logarithmic function for the spherical model as in Equation (16). This result corresponds to the *shape* of the variograms of the linear and spherical data models (Figure 7(a)) respectively. The impact of different spatial patterns on P_c and E_r is described in detail in Section 5.4.

5. MEASUREMENT AND CORRELATION MODEL

The patterns and levels of spatial correlation observed in the measured environmental data can give us an insight on the potential benefit of the CAG algorithm deployed in the real world. Eventually, the spatial data models can be used for predicting near future data or missing data.

In Section 5.1, we present the common variogram models used to classify spatially correlated data. In Section 5.2, we describe our setup for measuring the environmental data. Our fine granularity (10 meter internode distance) dataset is a significant improvement over the coarse granularity (tens of kilometers internode distance) dataset used by existing studies on spatial correlation in the wireless sensor networks [Deshpande et al. 2004; Jindal and Psounis 2004; Pattem et al. 2004]. In Section 5.3, we present the results from analyzing our collected data using spatial statistical techniques. In Section 5.4, we studied the performance and accuracy of CAG with the data model derived from our empirical dataset.

5.1 Variogram Models

The semivariogram[4] is the most common way to characterize the correlation between pairs of points separated by a spatial distance [Dale et al. 2002]. In probabilistic notation, the variogram is defined as follows: $\gamma(h) = \frac{1}{2}E[(X(p) - X(p+h))^2]$ for all possible locations p, where $X(p)$ and $X(p+h)$ are the values at the head and tail of each pair of points at a distance h. Variograms with 0 slope present no correlation such as independently identically distributed random values. Under positive autocorrelation, points that are close in the (x, y) plane tend to have similar values of z, where z is the sensor value at location (x, y). Now we describe three common variogram models.

Spherical model: The variogram of a spherical model increases linearly in the beginning, then it becomes a sill, which is a plateau. That is, the expected difference of the sensed values between two points stops increasing at a certain point although the distance (h) between the nodes increases. In a spherical pattern, data is correlated over a shorter distance than in linear or fractal patterns.

[4]We simply refer the semivariogram as a variogram from now on. The difference between the two is the multiplicative factor of 2 which only affects the magnitude and not the trend of the statistics.

(a) Picture of Exposition Park in Los Angeles.

(b) 4th floor of Tutor Hall at USC.

(c) Map of Great Duck Island with sensor deployment.

Fig. 4. Pictures of outdoor and indoor environments where the data is measured, and the map of the Great Duck Island.

Linear model: Positive correlation in a linear model stretches over a longer distance than in other correlation data models. In this model, data becomes less correlated as distance increases; this relation continues without stopping. We would say that a linear model is a stronger correlation model than other models, in terms of distance.

Fractal model: Fractal structures are ubiquitous in nature, with the key property of self-similarity across a range of spatials scales [Halley et al. 2004]. Fractal objects or behaviors often emerge in ecological models even if the models are not explicitly designed. Natural landscapes are not ideal fractals, but such models provide the simplest available means of simulating spatially complex landscapes, and serving as neutral habitat models. For a fractal pattern on a two-dimensional landscape, the relationship should be linear in a graph of $log(\gamma(h))$, against the $\log(h)$, with the measurement points stretching at least 2 orders of magnitude on each axis [Halley et al. 2004].

The Equations for the spherical and linear models used for variograms are given below, where c is the nugget effect ($\gamma(0) = c$), S is the sill, and a is the range of influence [Schabenberger and Gotway 2005]. As the distance h approaches zero, the measurement error and microscale variation induce the nugget effect.

$$\gamma(h) = \begin{cases} c + (S-c)\{\frac{3}{2}\frac{h}{a} - \frac{1}{2}(\frac{h}{a})^3\}, & \text{for } 0 < h \leq a \\ c + (S-c), & \text{for } h \geq a \\ 0, & \text{otherwise} \end{cases} \quad : \text{for spherical}$$

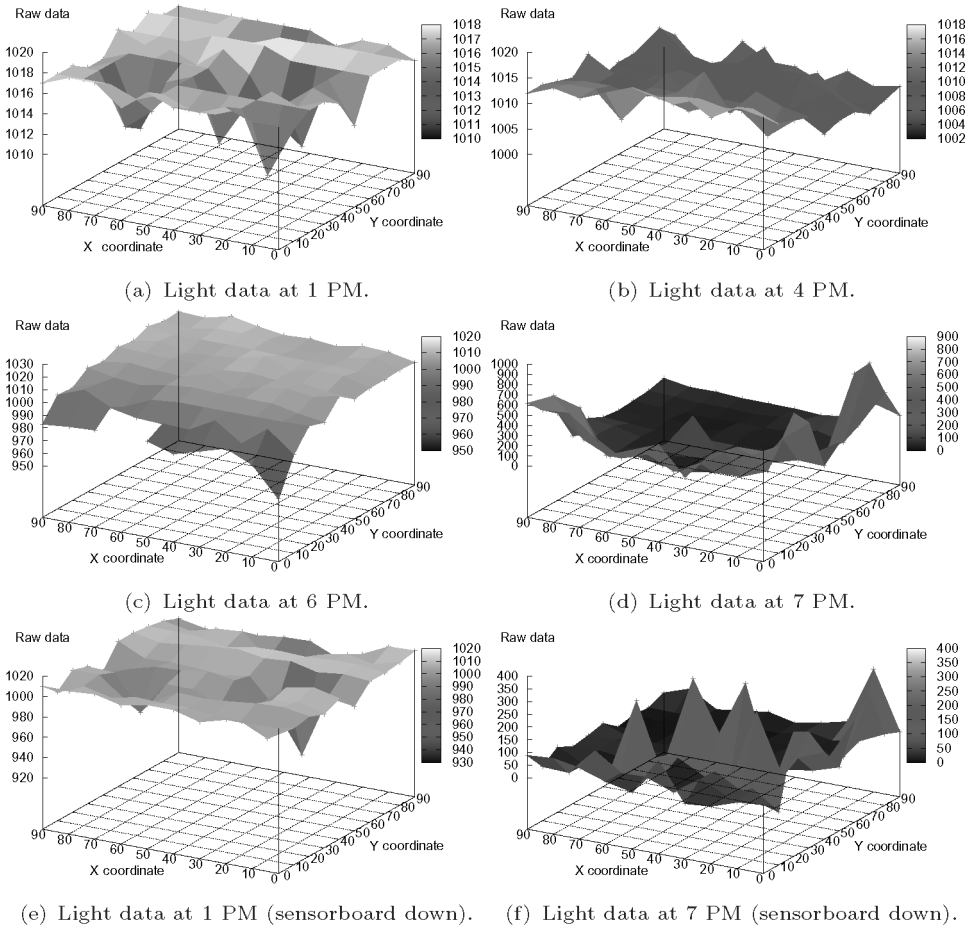
$$\gamma(h) = c + (S-c)\frac{h}{a}. \quad : \text{for linear}$$

These models are common for variograms used in practice in spatial statistics [Schabenberger and Gotway 2005].

5.2 Data Sets

For modeling, analysis, and simulation, we used the following four data sets, which we describe below.

5.2.1 *Sensor Data Measurement in a Regular Grid.* We measured two modalities (light and temperature) of real data using mica2 motes and MTS



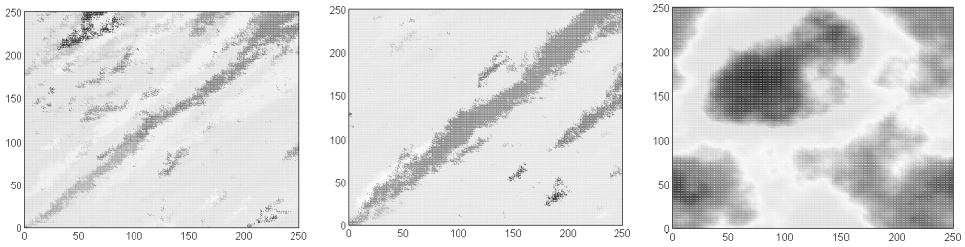
(a) Light data at 1 PM. (b) Light data at 4 PM.

(c) Light data at 6 PM. (d) Light data at 7 PM.

(e) Light data at 1 PM (sensorboard down). (f) Light data at 7 PM (sensorboard down).

Fig. 5. Measured light data from Exposition Park. The y-axis is the ADC value of the sensor reading. All the sensorboards are facing the sky except (e) and (f) in which case the sensorboards are facing down.

300 sensor boards in two different environments, outdoor at Exposition Park in Los Angeles (Figure 4(a)) and indoor on the 4th floor of Tutor Hall at USC (Figure 4(b)). At each position, the mote and the sensor board was placed on the ground and data was measured twice to understand the sensitivity of the data depending upon the sensor board's orientation: (1) the sensor board facing the sky, and (2) the sensor board facing the ground. Samples were taken at 200 millisecond interval. The arithmetic mean of twenty values was used as each reading to prevent getting inaccurate data that might be generated from a single instance of a malfunctioning sensor. It was not our goal to model or incorporate sensor errors into our protocol. For our measurements, a mote (directly connected to a laptop) was quickly repositioned to the next position on the grid after a set of measurements. The delay is not significant compared to the time scales at which changes in the measured phenomena occur, and hence can be ignored. All our experiments are based on the raw sensor values.



(a) $7h$ data. (b) $9h$ data. (c) Spatial pattern data.

Fig. 6. Synthetic data using a statistical model ($7h$ and $9h$) and an ecological model (spatial pattern). The color of each pixel represents the magnitude of the sensor reading for that coordinate.

A small subset of the light measurements from Exposition Park is presented in Figure 5.[5]

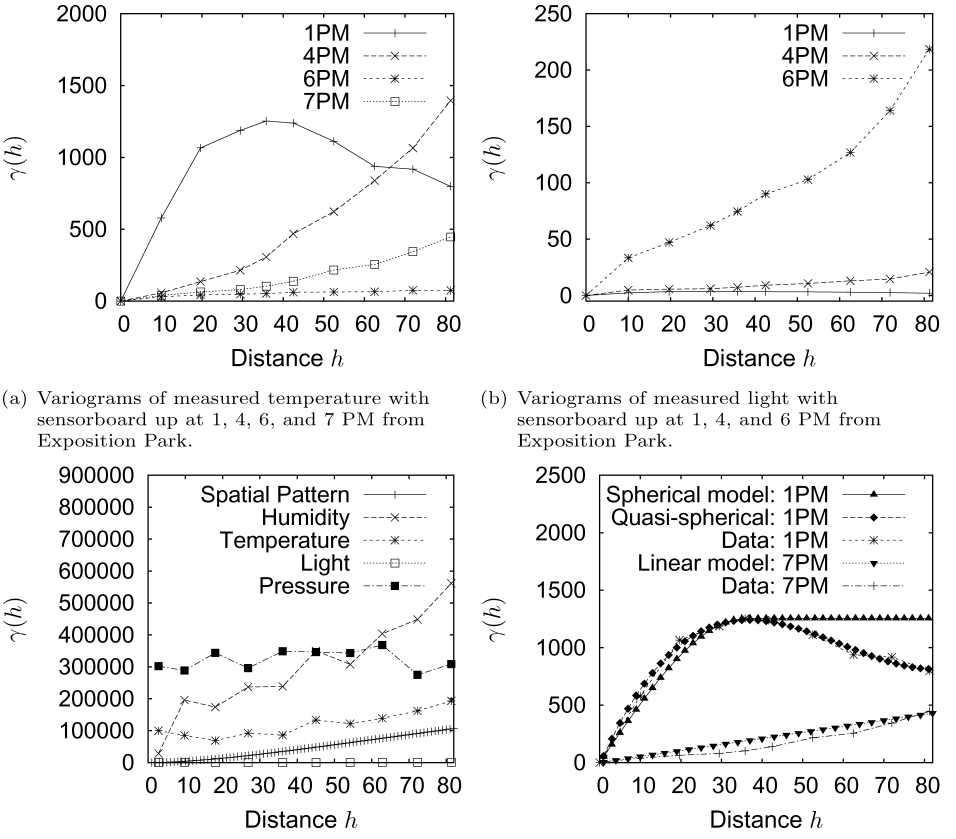
—**Outdoor environment (Exposition Park in Los Angeles):** We measured the environmental data at 100 (10m $\times$ 10m grid) positions in Exposition Park located in Los Angeles. Light and temperature readings were taken at four different times (1, 4, 6, 7 PM) of the day. We decided upon 10 meters as an internode distance because less than 10 meters is too redundant to monitor the outdoor environment. Also, the mica2 mote radio (433 MHz Chipcon CC1000) is reliable up to about 35 meters even with the default transmission power from our measurement study [Yoon and Shahabi 2005a].

—**Indoor environment (4th floor of Tutor Hall at USC):** We took measurements at 40 locations with a 5 meter internode distance in the rooms and hallway on the 4th floor of Tutor Hall at USC at 7 PM. We deployed the sensors more densely than in the outdoor environment because the data is affected by various indoor equipment (desks, chairs) and building structure (walls, windows). These factors can also weaken the radio transmission.

5.2.2 *Data with Irregular Mote Placement on Great Duck Island.* The four modalities (humidity, temperature, light, and pressure) measured on Great Duck Island [Mainwaring et al. 2005] constitute this data set (Figure 4(c)). Different modalities are in different units, but we used raw values in all cases. As these sensor nodes are not deployed in a grid (irregular internode distance), distances are subdivided into a number of intervals called *lags* to simplify variogram computation [Dale et al. 2002].

5.2.3 *Synthetic Sata from the Statistical Model.* Sensor data is generated using the method suggested in Jindal and Psounis [2004] for a 250m $\times$ 250m grid. Five data sets with different degrees of correlation are generated with parameters $\alpha = 1/2^i$, β, and $h = 1, 3, 5, 7$, and 9. The correlation coefficient h determines the level of correlation; an h of 1 generates data with almost no spatial correlation (similar to independently identically distributed random), and a larger h results in a higher spatial correlation. Our previous research [Yoon

[5]Here we do not show all the data we measured due to space constraints.



(a) Variograms of measured temperature with sensorboard up at 1, 4, 6, and 7 PM from Exposition Park.

(b) Variograms of measured light with sensorboard up at 1, 4, and 6 PM from Exposition Park.

(c) Variograms of spatial pattern and the measured data (light, temperature, humidity, pressure) from Great Duck Island.

(d) Variograms of temperature data from Exposition Park and their models in spatial statistics: spherical and quasi-spherical (1 PM) and linear (7 PM) models.

Fig. 7. Variograms of measured and synthetic data.

and Shahabi 2005b] used this data to compare the effects of different spatial correlation levels on the efficiency and accuracy.

In this article we compare P_c as given in Equations (13) and (14) using empirical data, and $7h$ (Figure 6(a)) and $9h$ (Figure 6(b)) using synthetic data.[6]

5.2.4 *Synthetic Data from the Ecological Model.* We used the spatially correlated data (Figure 6(c)) generated based on the ecological (environmental) patterns model provided by Lennon [2000] in 250m $\times$ 250m grid. Even though this data is synthetic, it contains realistic spatial patterns with known spatial properties similar to the fractal pattern found in the environment.

This synthetic pattern is generated by using stochastic noise. We generated a power spectrum with $\frac{1}{f^2}$ frequency using the normal error distribution in two dimensions, and then an inverse Fourier transform, to obtain the spatial pattern;

[6]In Yoon and Shahabi [2005b], we reported the details on the different levels of spatially correlated data, and the corresponding results.

this technique is used to create many natural looking fractal forms [Lennon 2000]. We generated the scale invariant, brown noise (which is strictly quasi fractal), with the fractal dimension $D = 2$.

Figure 7(c) includes the variogram of this pattern. This spatial pattern presents the fractal characteristic of the environment with a high correlation level between $7h$ (Figure 6(a)) and $9h$ (Figure 6(b)) [Yoon and Shahabi 2005b].

5.3 The Spatial Data Model

In this section, we mathematically model the property of spatial correlation using the measured sensor data from different environments. We observed that the linear correlation model (which can potentially be a fractal model) and the spherical data model were prevalent in the environmental data we measured.

Figure 7(a) shows the variograms of temperature data measured in Exposition Park. All the temperature data from different times of the day show a linear property (6 PM shows almost random) except for the 1 PM data, which shows spherical characteristics. Figure 7(b) shows the variograms of light values measured in Exposition Park. Each variogram from a different time of the day shows different linear functionality; 1 PM shows the least correlation (almost random), and 6 PM shows the strongest correlation.

Because of the strong sunlight at 1 PM, the temperature values in the outdoor environment changes by a large amount (480 to 660) even at a short distance, due to the shade under the trees [Yoon and Shahabi 2005a]. On the other hand, the range of light readings at 1 PM is small (1010 to 1018) under the same shadows (almost random as shown in Figure 5(a)). The magnitude of the variogram for light at 7 PM is much larger than those at 1, 4, and 6 PM because of its huge data range (0 to 900) from the effects of street light (some places were bright and some were completely dark) at night (Figure 5(d)), so it is not included in Figure 7(b).

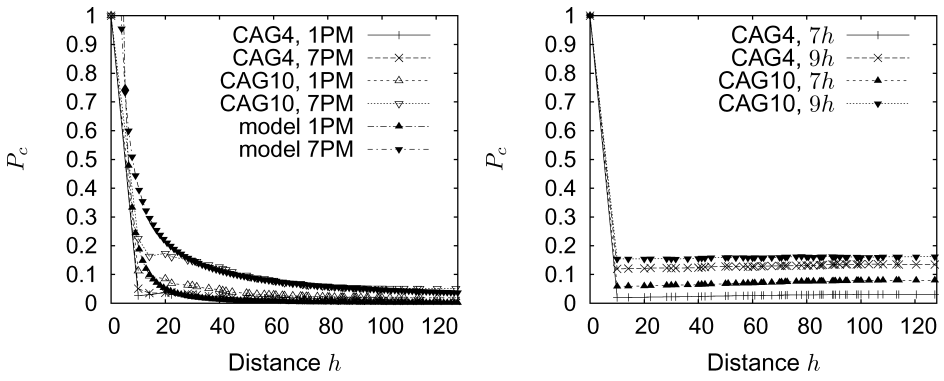
The spatial correlation property holds for both the face-up and face-down orientations of the sensorboard. For the light sensor values the sensed values with sensorboards facing up capture the physical phenomenon better, and show the spatial correlation more clearly than with sensorboards facing down. On the other hand, the temperature value was not sensitive to the orientation of the sensorboards.

When we apply the temperature data to the correlation data model equations, we get Equation (10) for the spherical model and Equation (12) for the linear model. For a finer match with the data at 1 PM, we build a quasi-spherical model using a polynomial equation by regression, as in Equation (11).[7]

$$\gamma(h) = \begin{cases} 1254 \times \left(\frac{3}{2}\frac{h}{36} - \frac{1}{2}\left(\frac{h}{36}\right)^3\right), & \text{for } 0 < h \leq 36 \quad \text{: for spherical} \\ 1254, & \text{for } h \geq 36 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

$$\gamma(h) = 0.0089h^3 - 1.5906h^2 + 80.76h - 20.594 \quad \text{: for quasi-spherical} \tag{11}$$

[7]We model the data mathematically for the purposes of our study. A more sophisticated modeling is out of the scope of this study. We will address this problem in our future work.



(a) Simulation and mathematical model of P_c with CAG4 and CAG10 with temperature data from Exposition Park. Linear model (model 7PM) has larger heavy tail than spherical model (model 1PM).

(b) Simulation model of P_c with CAG4 and CAG10 using $7h$ and $9h$ synthetic data.

Fig. 8. Mathematical model of P_c. *CAGn* means CAG with $\tau = n\%$. Both (a) and (b) use 10 meter internode distance in a regular grid.

$$\gamma(h) = \frac{425}{81} \times h. \qquad \text{: for linear} \tag{12}$$

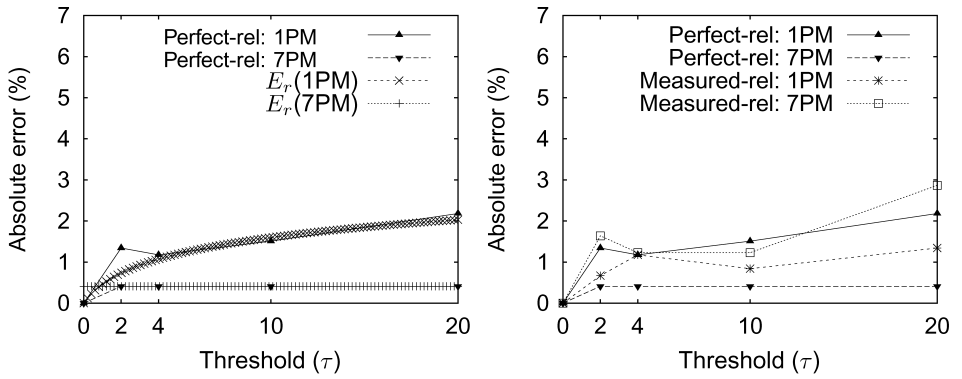
As shown in Figure 7(c), variograms of four modalities measured on Great Duck Island and a variogram of spatial patterns from the ecological model show the linear function. If we ignore the nugget effect [Schabenberger and Gotway 2005], which makes the variogram start at a non-zero value at $h = 0$, all variograms in Figure 7(c) become similar in magnitude and pattern. Note that the important factor in the variogram is not the magnitude, but the *shape* of graph. The different magnitudes of each modality in the variogram are due to the differences in magnitudes of the raw sensor values across different modalities.

Most variograms using the real sensor data from Exposition Park (Figure 7(a), 7(b)), Tutor Hall (excluded for brevity), and Great Duck Island (Figure 7(c)) show the linear pattern (except for the 1 PM temperature from Exposition Park, which shows the spherical pattern) similar to that of ecological spatial patterns (Figure 7(c)) in different magnitudes. Thus, we conclude that linear and spherical correlation models are prevalent in our sensor readings. We use these two models to analyze and evaluate CAG's performance and accuracy in the next section.

5.4 The Impact of Different Correlation Models on CAG

While we proved the analytic lower bound of efficiency and accuracy of CAG with uniform distribution of sensor data in Section 4, in this section, we investigate the impact of different correlation patterns observed in the measured environmental sensor data on the efficiency and accuracy of the CAG algorithm. This analysis can give us insight into how much energy saving and accuracy CAG can achieve in the real world.

We now analyze the performance of CAG based on the real measured data. We model P_c, the probability of two nodes being in the same cluster, empirically



(a) Simulation (streaming mode with perfect reliability) and analytical models of the absolute error for two temperature data patterns (linear for 7 PM and spherical for 1 PM).

(b) Absolute error of two data patterns (linear for 7 PM and spherical for 1 PM) of temperature in terms of threshold both with perfect reliability and the default transmission power.

Fig. 9. Accuracy model as a function of threshold (τ) and internode distance (h).

as the level of correlation using PDF as a function of internode distance h and threshold τ for each spatial data model. We observed that the CAG can achieve better efficiency and accuracy with the linear pattern than with the spherical pattern.

We selected two data sets as the representative environmental data models; the temperature data at 1 PM, and 7 PM measured in the Exposition Park. We model P_c by curve fitting the simulation result of CAG using $\tau = 4\%$ and 10%. Figure 8(a) shows the result of simulations and their mathematical models.[8]

For small τ (e.g., CAG with $\tau = 4\%$ or less), we observed that both spherical and linear patterns present similar P_c: both patterns follow the same polynomial, which is a function of h and τ, as in Equation (14). For large τ (e.g., CAG with $\tau = 10\%$), we observed that the P_c of the linear model is more heavy-tailed than that of the spherical model; P_c depends on the data model, h, and τ. Thus, we model P_c for relatively large τ, such as the linear pattern with a logarithmic function as presented in Equation (13), and the spherical pattern with a polynomial function as in Equation (14).

$$P_c = \frac{\tau}{2(\log(h) + h)} \quad \text{: for linear,} \tag{13}$$

$$P_c = 2\tau h^{-2} \quad \text{: for spherical.} \tag{14}$$

In our earlier work [Yoon and Shahabi 2005b], we observed P_c larger than those in the Figure 8(a) with the synthetic $7h$ data, which corresponds to the similar correlation level of temperatures from Great Duck Island. This difference can be attributed to the changed measurement setting: in this study we used 10 meter internode distance in a regular grid while in our earlier

[8]We did not measure the sensor data where $D_{ij} < 10$m. Thus, we just connected two measured points where $D_{ij} = 0$m and 10m.

work [Yoon and Shahabi 2005b], we used random positioning with an average internode distance of 10 meters. To help us compare these two sets of results under the same conditions we repeated experiments from our previous work with nodes placed in a regular grid with 10 meter internode distance using $7h$ and $9h$ data. We observed that our measured temperature from Exposition Park (Figure 8(a)) is more spatially correlated than $7h$ synthetic data (Figure 8(b)) in terms of P_c.

The absolute error of the streaming mode of CAG, E_r, is obtained empirically by the curve fitting graphs in Figure 9(a). We model E_r as a constant for a linear model in Equation (15) and logarithmic function for a spherical model in Equation (16).

$$\begin{aligned} E_r &= 0.41 \\ &= c, \quad \text{where } c \text{ is a constant} \qquad \text{: for linear,} \qquad (15) \\ E_r &= \frac{3}{2} log(\tau + 1) \qquad \text{: for spherical.} \qquad (16) \end{aligned}$$

We demonstrate that E_r is bounded by the given threshold in the streaming mode even under the packet loss using the mica2 radio profile. Figure 9(b) shows that even with the packet loss, the absolute error is always bounded by τ. We observed that the E_r of the linear pattern (7 PM) is smaller than that of the spherical pattern (1 PM) at a given threshold without packet loss. However, with the packet loss, we observed that E_r for 7 PM is not always smaller than that of 1 PM, as shown in Figure 9(b).

6. EXPERIMENTAL STUDY

In this section, we describe 1) the evaluation metrics and experimental setup, 2) results from the evaluation of the interactive mode, and 3) evaluation results of the streaming mode, which supports cluster adjustment.

6.1 Evaluation Metrics and Experimental Setup

The primary metric used for evaluation, the *reduced number of transmissions*, uses the number of transmissions to estimate the energy cost in WSN. This approach is reasonable because radio transmissions consume far more energy than any other operation in a sensor node [Hill et al. 2000]. In the interactive mode, the reduced number of transmissions is calculated as $\frac{nTX(TAG)-nTX(CAG)}{nTX(TAG)} \times 100$, where $nTX(TAG)$ and $nTX(CAG)$ are the number of transmissions for TAG and CAG respectively.

In the streaming mode, we compute the *number of transmissions* to compare overheads between CAG and TAG. This metric is also used to compare the breakdown of each transmission overhead.

In our calculations, nTX excludes the query packets because this number is the same in both TAG (one-shot or streaming) and CAG (interactive or streaming) regardless of τ. In either mode of both protocols, there is a single instance of query dissemination.

Another metric is the *accuracy of result* as an *absolute error* for a given τ calculated as $|Estimated\,Result - Correct\,Result| \times 100$. Note that the absolute

error is computed using $Estimated\,Result$ and $Correct\,Result$ from the same query cycle.

We compute the average number of bridge nodes in the data forwarding paths to understand their contribution to the total communication overhead. Bridge nodes are used to keep the topology connected and they can optionally contribute to the aggregate. Note that the bridge nodes did not contribute to the aggregate in our simulations.

We explored the impact of 3 different densities (moderate, dense, and sparse) on the reduced number of transmissions. For moderate density, we randomly placed 375 nodes in a 250m $\times$ 250m grid, which results in at least a 5-hop topology with each node having an average of 17 neighbors. We randomly placed 550 nodes (average 26 neighbors per node) in dense deployment, and randomly placed 200 nodes (average 9 neighbors per node) in sparse deployment with other conditions constant.

To understand the effect of packet loss on accuracy, we ran simulations on two types of topologies: 1) lossless topologies and 2) topologies constructed using empirical loss rates. Comparing results from these two topologies gives us insight into the difference between theoretical results and what might be observed in real world implementation of CAG. We used the loss profile from our own measurement using mica2 motes to assign reliabilities to links between nodes. The reception rate was measured using a 433 MHz Chipcon CC1000 by counting the successfully received packets among 500 packet transmissions at the default radio transmission power. We also configured CAG and TAG not to use retransmissions for our experiments. Table III describes the metrics used to evaluate the CAG.

We used the TOSSIM simulator of TinyOS 1.1.8 for our simulation study [Levis et al. 2003]. We used the temperature data collected with sensorboard up at 1, 4, 6, and 7 PM from Exposition Park. One hundred nodes were deployed in a regular 100m $\times$ 100m grid. We also used temperature readings collected on Great Duck Island to study the efficiency of CAG with a long temporal history (from 35 nodes every hour for four days). Each node at position (x, y) uses the value from the corresponding position in the empirical data sets.

We generated four different topologies (the root node is placed at each corner of the 100m $\times$ 100m grid) with the above configuration and averaged results over 20 runs for each topology. We observed that the inclusion of the bridge nodes in the aggregation only contributes marginally to the improvement of the precision of the result. We chose $\tau = 0, 2, 4, 10$ and 20% to cover typical values of τ that users might be interested in.

6.2 Results

In this section, we report the results from our experiments both in interactive and streaming modes. Our key finding is that CAG in the interactive mode produces results with very small and often bounded errors with dramatically reduced message overhead compared to TAG. In the streaming mode, efficiency compared to TAG is even higher while at the same time guaranteeing that the errors in the results are always bounded by the user-provided threshold. In this

section, we only report the results for the AVG operator. We expect qualitatively similar results for other aggregation operations. We omit those results due to lack of space.

6.2.1 *Interactive Mode of CAG.* In this Section, we present the performance and precision tradeoff in the interactive mode of CAG. In this mode, the clusterhead values are not weighted by the cluster sizes. Thus, if the data from all the nodes do not follow the normal distribution, we cannot guarantee that the error from the result is bounded by the given τ. Figure 10 shows a snapshot of the CAG tree with 375 nodes randomly placed in 250m $\times$ 250m space with $9h$ synthetic data and $\tau = 20\%$. The root node (the big black square) is located in the bottom-left area, and all other nodes shown in black circles are in the same cluster with the root nodes. As expected, most new clusters are built along the diagonal band spanning from the top-left to the bottom-right (clusterhead nodes, small black square nodes, are spread there), where there is a large difference in the magnitude of data. This figure confirms the validity of our implementation of the CAG algorithm.

Figure 11(a) shows the improved performance of CAG compared to TAG in terms of the reduced number of transmissions in the interactive mode using the default transmission power. As we mentioned earlier, we compare the CAG interactive mode against TAG with one-shot query because CAG interactive mode also uses one-shot query. CAG with $\tau = 4\%$ has a transmission savings up to 37.5% at 6 PM and CAG with $\tau = 10\%$ has a transmission savings up to 51.25% at 7 PM. An interesting result is that CAG can save 8.75 to 20% even with $\tau = 0$, which guarantees that the results have no error. This is because nodes with the identical sensor readings form clusters, thereby allowing no error.

Figure 11(b) shows the number of bridge nodes. As the error threshold increases, more nodes choose not to respond. Thus, we need more bridge nodes to keep the tree connected. Because a modest increase in the number of bridge nodes is accompanied by a dramatic reduction in the number of clusterheads, the total number of transmissions still decreases as the threshold value increases.

Figure 11(c) shows the accuracy of the result obtained using the interactive mode of CAG with the empirical radio profile. Due to the unreliable links, the resulting errors are out of bound when $\tau = 2\%$ at 7 PM. The error caused by the packet loss is 9.375% when $\tau = 10\%$ at 4 PM.

Figure 11(d) shows the accuracy of the result obtained using the interactive mode of CAG with perfect link reliability. Even though we expected that the resulting error is not guaranteed to be within the given threshold τ in the interactive mode, we observed that the resulting error was always bounded by τ. This can be an indication that our measured temperature data in the physical world follows the normal distribution.

We investigated the impact of three different densities (moderate, dense, and sparse) on the reduced number of transmissions. With a denser node deployment, CAG saves more transmissions by exploiting the increased correlation in readings from nearby sensors. Sparse deployment results in weaker data

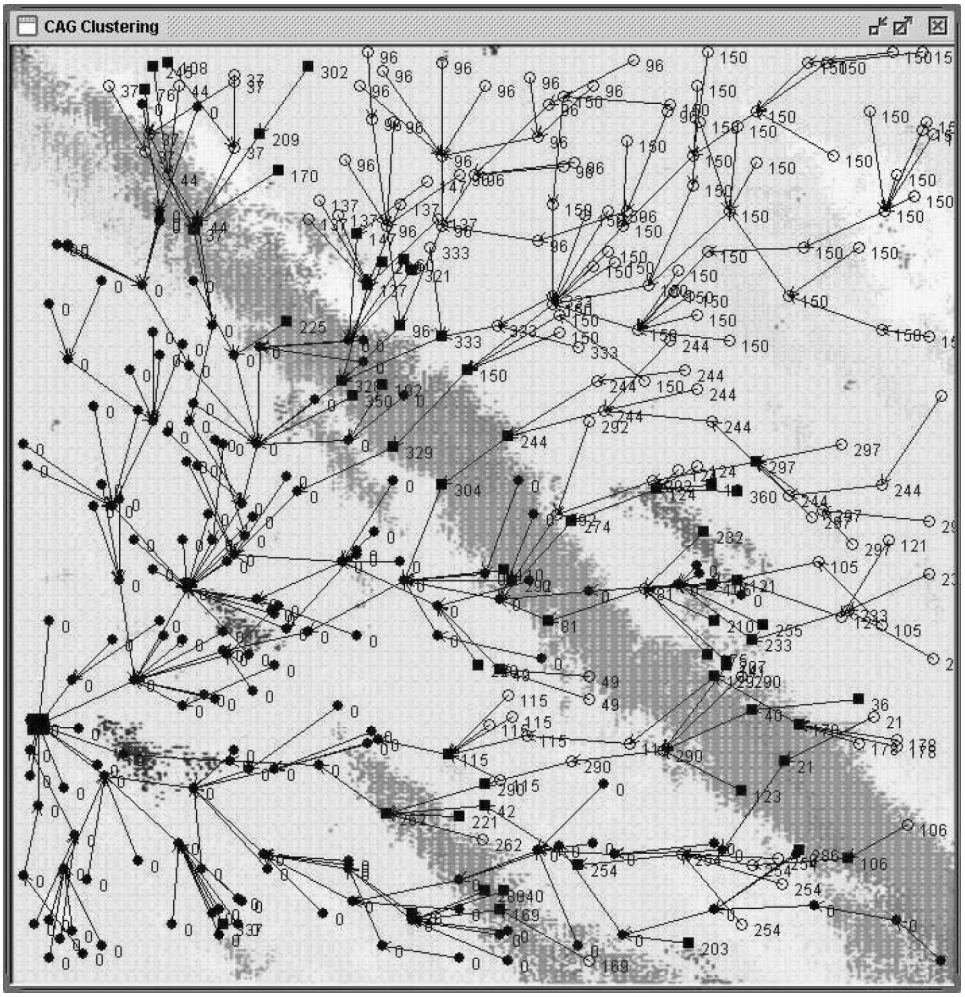
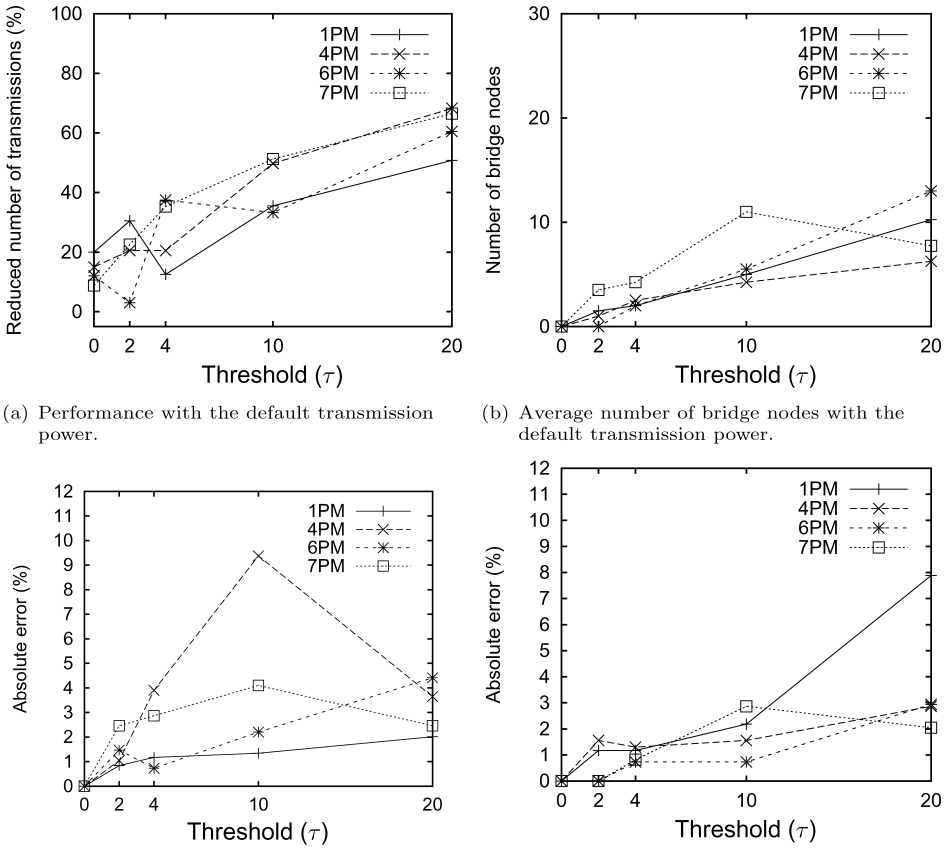


Fig. 10. A snapshot of the CAG tree with 375 nodes randomly placed in 250m × 250m space with $9h$ synthetic data and $\tau = 20\%$. The big black square near the bottom left corner is the root node and the other small black circles are nodes in the root cluster. Clusterhead nodes (except the root node) are the small black squares and the non-clusterhead nodes (except the root cluster) are small empty circles. The number beside each node indicates clusterhead node id, and the arrow points to the parent node in the query routing tree.

correlation, which increases the transmission overhead for CAG. We omit the figure presented in our earlier paper [Yoon and Shahabi 2005b] due to the space constraint.

6.2.2 *Streaming Mode of CAG.* In this section, we present the impact of data dynamics (both spatial and temporal) on the performance and accuracy of the streaming mode of CAG. In our evaluations, we systematically compare 1) the total transmission overhead and 2) the cluster adjustment overhead in the streaming mode with alternative approaches, while presenting 3) the detailed



(a) Performance with the default transmission power.

(b) Average number of bridge nodes with the default transmission power.

(c) Precision with the default transmission power.

(d) Precision with perfect link reliability (bounded by τ).

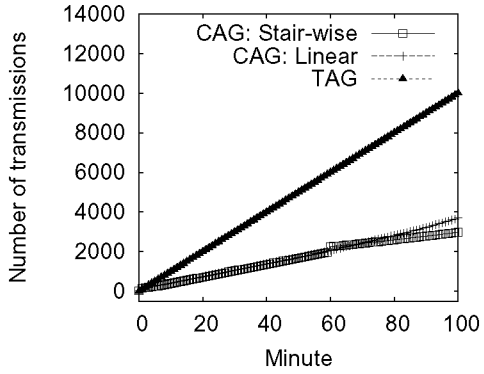
Fig. 11. Performance and precision tradeoff in the interactive mode with measured temperature data from Exposition Park.

breakdown of the total message overhead. Our main result from studying the streaming mode of CAG is that small and bounded error is achievable while significantly reducing the message overhead compared to TAG.

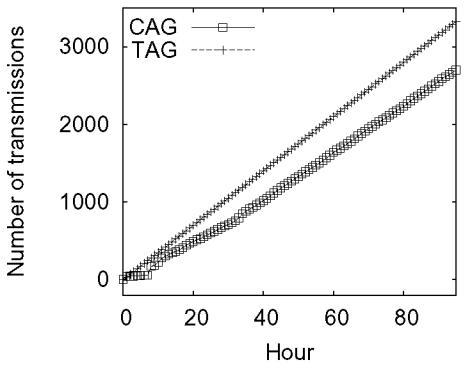
We used three different data sets for our measurement study in this section: Great Duck Island dataset, stair-wise dataset, and linear dataset.

Great Duck Island dataset: We obtained the coarse granularity (recorded once per hour) time series temperature data from the measurements conducted on Great Duck Island from 35 nodes for four consecutive days (Figure 12(f)). This data set gives us insight into the long term performance of CAG with high data dynamics.

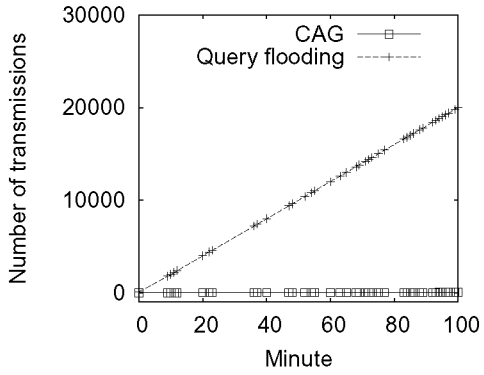
In order to investigate the impact of different temporal patterns of data on different aspects of CAG performance, we first analyzed the data from our measurements in Exposition Park and that from Great Duck Island. We observed two common data patterns in our analysis of data from Great Duck Island—(1) a stair-wise pattern in which data do not change for a long period of time



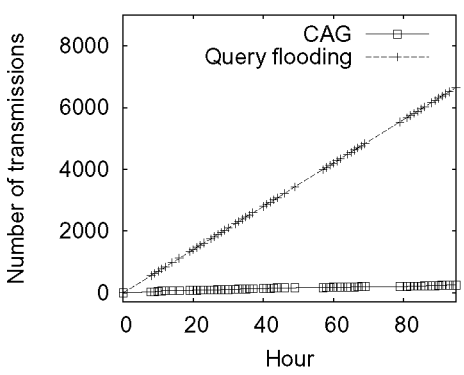
(a) Comparison of the total transmission overhead (accumulated) between TAG and CAG using the linear and stair-wise datasets with $\tau = 20\%$.



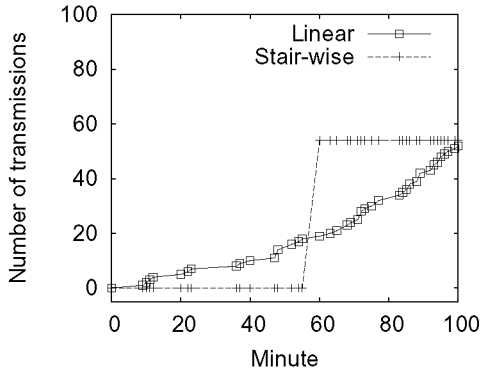
(b) Comparison of the total transmission overhead (accumulated) between TAG and CAG using the Great Duck Island dataset with $\tau = 20\%$.



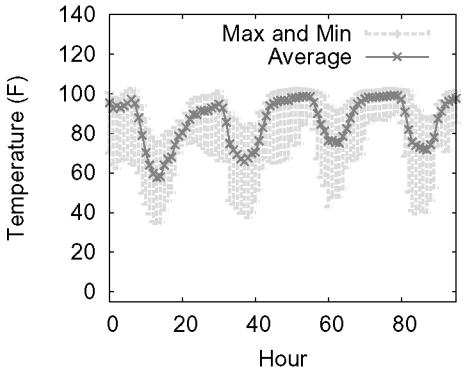
(c) Comparison of cluster adjustment overhead (accumulated) using the linear dataset with $\tau = 20\%$.



(d) Comparison of cluster adjustment overhead (accumulated) using the Great Duck Island dataset with $\tau = 20\%$.



(e) Comparison of cluster adjustment overhead (accumulated) between linear and stair-wise datasets using with $\tau = 20\%$.



(f) Maximum, minimum and average temperature of the Great Duck Island dataset.

Fig. 12. Impacts of different spatial and temporal correlations on the performance of CAG. All experiments use the radio profile for default transmission power.

and change drastically whenever there is a change, and (2) a linear pattern in which data changes monotonically, increasing or decreasing. The stair-wise pattern is due to the shadows that keep certain places cool for a period of time even in the middle of the day, resulting in the sharp changes of temperature.

Stair-wise dataset: The stair-wise dataset for our evaluation is taken from the measurements from Exposition Park. The temperature readings between 4 PM and 6 PM are interpolated as stair-wise data: no change from 4 to 5 PM, at 5 PM sudden change to the 6 PM level, and no change from 5 to 6 PM. This dataset has one-minute resolution in the time axis.

Linear dataset: We do not have access to linear time series data at a high resolution from direct sensing in large deployment. Instead, we generate a linear dataset by linearly interpolating (one minute intervals) the temperature snapshots taken at Exposition Park at 4 PM and 6 PM, described in Section 5.2.1.

Simulation with these finer granularity data sets (linear and stair-wise) can help us understand CAG's performance with high data correlation at a smaller time scale.

Now we proceed to present the main results from our simulation study. The first observation we made is that the CAG is much more efficient than TAG in terms of the total number of messages (control and data forwarding) incurred by both the algorithms. Figure 12(a) shows that CAG with linear dataset is able to achieve 63.07% reduction in message overhead. With a stair-wise data pattern, CAG used 70.24% fewer messages to compute the aggregate result. We also compared the message overhead of CAG and TAG with the Great Duck Island dataset and found that CAG consistently incurred about 19% less overhead than TAG (Figure 12(b)). We further note that these reductions in message overhead were achieved while incurring a small error in the result (Figure 14). We attribute this reduction in number of transmissions to the reduced number of nodes that send their responses to the root. In TAG all the nodes send their results to the root while in CAG only the clusterheads transmit and aggregate the results.

We now justify the need for the cluster adjustment algorithm in the streaming mode using experimental evaluation. Without reclustering, the clusters can quickly become inconsistent as new sensor readings no longer fall within the clustering range. One approach that addresses this problem is called *reclustering using query flooding*. This technique floods the query from the sink periodically, thus recreating the query routing tree and all the clusters from scratch, using the latest sensor readings. The second approach is using a separate reclustering algorithm like the one we developed for CAG. With our algorithm, we repair the clusters that need to change and avoid changing and adjusting the clusters that have nodes with sensor readings still within the clustering range. Figure 12(c) shows that the cluster adjustment overhead, with linear dataset, remains fairly constant and reaches a maximum of 52 while the reclustering overhead using query flooding reaches 20,000 messages in 100 minutes even though both the algorithms are reclustering at the same frequency. Figure 12(d) shows the same result using the Great Duck Island dataset. In this figure, the reclustering overhead using query flooding reaches 6650 messages while that of CAG reaches 249 messages in 96 hours. Thus, we conclude that although

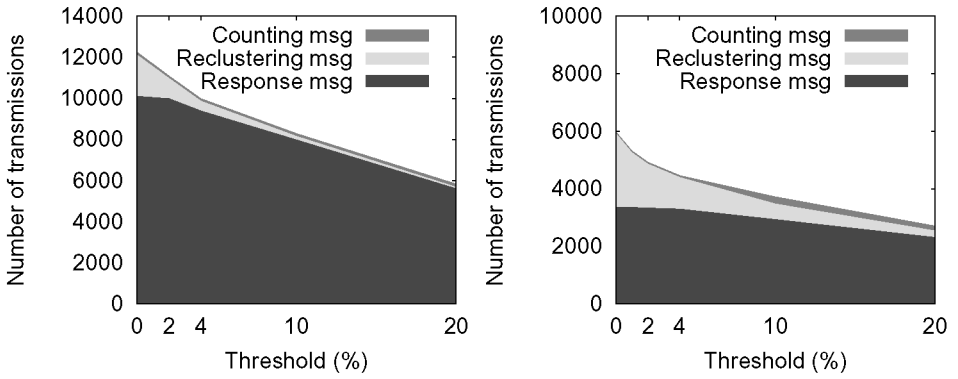
reclustering using query flooding works, it has an unacceptable overhead if one desires to use this technique at a frequency high enough to ensure the consistency of cluster membership. On the other hand, the CAG reclustering algorithm is able to run the adjustment algorithm at the same frequency with much smaller overhead. The CAG adjustment algorithm achieves this efficiency by using local repair and avoiding a global adjustment of all the clusters.

We also investigated how the cluster adjustment overhead evolves over time for different data patterns. Our experiments suggest that the message overhead for linear and stair-wise data is similar though they evolve differently. We described earlier that both the linear and stair-wise datasets are constructed using the temperature snapshot at 4 PM and 6 PM. Figure 12(e) shows that with the stair-wise dataset, the adjustment overhead remains 0 until the data jumps from one value to the next. During that jump, there are a lot of nodes changing the clusters which result in a large number of adjustment messages transmitted in the network. Once reclustering is done in response to this jump, there is no more need for reclustering and there is no further adjustment overhead. For the linear dataset, the reclustering process is continuous as there is a small change in data during the entire duration of the experiment. Even though instantaneous adjustment overheads are seen to be significantly different for the two datasets, we note that accumulated message overhead for both the datasets is similar at the end of the experiment in 100 minutes.

Our next observation concludes that the cluster adjustment overhead becomes smaller with a larger user-provided threshold τ. In Figure 13(a), which shows results with the linear dataset, the adjustment overhead with $\tau = 2\%$ is 987 messages. The overhead gets smaller as a larger threshold is chosen, eventually yielding a total of 76 messages with a τ of 20%. With the Great Duck Island dataset, in Figure 13(b), the overhead starts at 1531 messages with $\tau = 2\%$ and becomes 229 messages with $\tau = 20\%$. This result can be explained by the fact that a smaller threshold corresponds to a smaller range of values allowed within the cluster. Even a small change in sensor reading can make the node fall outside the allowed range, which forces a node to find a new cluster. Hence a larger number of nodes participate in cluster adjustment, which results in a larger adjustment overhead with smaller thresholds.

We observe that the adjustment overhead is a much smaller fraction of the total overhead for the linear dataset (Figure 13(a)) compared to the Great Duck Island dataset (Figure 13(b)). Our linear dataset shows more correlation than the Great Duck Island dataset. Lower correlation results in frequent and large changes in sensor readings which necessitates frequent reclustering. This frequent reclustering explains the higher adjustment overhead with the Great Duck Island dataset compared to the linear dataset.

While not one of our main results, one interesting observation that we made during the course of our experiments is that as the threshold τ increases, the forwarding message overhead from the Great Duck Island dataset remains relatively flat (Figure 13(b)) compared to the gradual decrease in forwarding message overhead for the linear dataset (Figure 13(a)). We attribute this difference to different levels of correlation in the two datasets. The Great Duck



(a) Breakdown of transmission overhead using the linear dataset with default transmission power. Response message(bottom) overwhelms reclustering message(middle) and counting message (top).

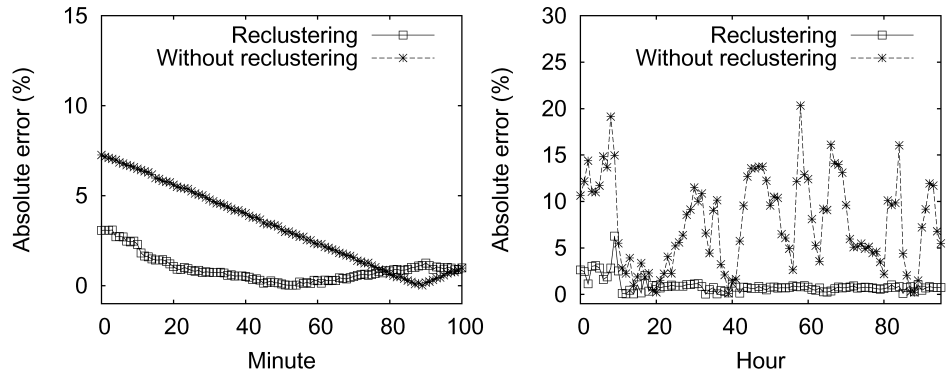
(b) Breakdown of transmission overhead using the Great Duck Island dataset with default transmission power. Response message(bottom) overwhelms reclustering message(middle) and counting message (top).

Fig. 13. Breakdown of transmission overhead with measured temperature data.

Island dataset is less correlated than the linear dataset because it spans a larger period of time (4 days vs. 2 hours). With both datasets, with small threshold, because of the small clustering range, a large number of clusters are formed. With the linear dataset, with a larger threshold, the data is more correlated, leading to large cluster sizes, which can be maintained over time. This results in lower data overhead, which is why we see a decreasing trend in forwarding overhead. With the Great Duck Island dataset, which has lower correlation, with a large threshold, even though large clusters are formed in the beginning, they are broken into smaller clusters because the data changes drastically over time. Thus, we end up with a large number of small clusters, which explains why the message overhead does not go down significantly even with a larger threshold.

Next we focus on the error in the results obtained using CAG. What relation is seen between the error in the result and the user-provided threshold? How does the accuracy of the result using CAG compare with the accurate result one could obtain using TAG? What improvement in accuracy, if any, is contributed by the adjustment algorithm?

The accuracy of result achieved indicates that the absolute error in the result obtained using CAG is always bounded by the user-provided threshold τ. With linear data, with a threshold of 20%, the error is always less than 3.09% compared to the accurate result (Figure 14(a)). With the Great Duck Island dataset, the error is always less than 6.26%, also with a threshold of 20% (Figure 14(b)). To understand the impact of CAG adjustment on accuracy, we ran the next set of experiments *without* the cluster adjustment algorithm. We expected the errors to be generally higher without the adjustment algorithm because sensor readings slowly drift away from the clustering range without periodic reclustering. The results in Figure 14(a) and 14(b) corroborate our expectation. The errors are not only generally higher without reclustering but also sometimes out of bound (Figure 14(b)).



(a) Accuracy with the linear dataset and $\tau = 20\%$. (b) Accuracy with the Great Duck Island dataset with $\tau = 20\%$.

Fig. 14. Accuracy results with CAG for different datasets with default transmission power.

We observed a downward trend in the absolute error in Figure 14(a). The sensor readings at 6 PM have smaller variance than data at 4 PM. In both, with and without reclustering, as time progresses towards 6 PM, the similarity of sensor reading at clusterhead and that at other nodes in the cluster increases. This continuous increase in similarity results in a downward trend in error for both cases (Figure 14(a)).

We omit the results on performance and precision using the indoor environmental data due to the space limitation and the fact that it does not show a significant difference from the results using the outdoor data. Our earlier work [Yoon and Shahabi 2005b] described the impact of different levels of correlation and density on efficiency and accuracy.

CAG allows a user to seek approximate answers to a query, thus enabling the network to conserve energy by reducing the number of transmissions. CAG, in general, is not expected to be used with very small thresholds because such thresholds prevent CAG from fully exploiting data correlation to significantly increase energy efficiency over TAG. We further observed that CAG provides bounded error all the time over all thresholds despite using mica2 radio profile (which is inherently unreliable) in our simulations. Based on our experimental evaluation, for data profiles expected to be similar to the one used in our simulations, we suggest that a $\tau \geq 4\%$ might be a reasonable trading point to achieve both small errors (bounded) and energy saving in reality.

7. CONCLUSION AND FUTURE WORK

In this article, we demonstrated the effectiveness of CAG, in both interactive and streaming modes, in performing energy efficient in-network aggregation leveraging both spatial and temporal correlations. We mathematically modeled the spatial correlation of the measured data and evaluated the efficiency and accuracy of CAG analytically and empirically. CAG can maximally take advantage of stronger spatial and temporal correlations by increasing energy efficiency and providing the results with predictable and bounded errors. These benefits are amplified as the density of nodes becomes higher. CAG is shown

to scale gracefully when the number of nodes in the network grows. Moreover, CAG is shown to be resilient to the packet loss. CAG is the first system that realizes the semantic broadcast to conserve energy, while ensuring bounded approximation by leveraging the spatial and temporal correlations prevalent in nature.

The streaming mode of CAG supports localized cluster adjustment and repair whenever the sensor readings are out of the clustering range. If a small number of nodes needs to join a different cluster (which happens frequently in our experiments due to a large change in sensor readings or communication problems), they can do so locally by communicating with the neighboring nodes associated with other clusters. By avoiding global communications and adjusting clusters only using local communications, CAG can save a significant number of transmissions and hence, energy, especially when only a few nodes change clusters.

We would like to extend this work by focusing on the design of a hybrid (proactive and reactive) clustering protocol. Current CAG implementation proactively collects the sensor reading per cluster in every epoch. Whenever the cluster structure changes, CAG will resample, recompute, and send back the aggregate value to the base station. This mechanism enables a reactive data acquisition. We would like to provide proactive and reactive data acquisition, depending on the application requirement or data characteristics.

ACKNOWLEDGMENTS

The authors would like to thank professor David Kempe for discussions about the updated algorithm, professor Bhaskar Krishnamachari for useful discussions, and Sundeep Pattem for valuable feedback. The authors also would like to thank professor Jonathan Yearsley for the spatial data from the ecological model, professor John Heidemann and professor Wei Ye for useful comments, and the anonymous reviewers for their detailed comments on how to improve the article.

REFERENCES

CONSIDINE, J., LI, F., KOLLIOS, G., AND BYERS, J. 2004. Approximate aggregation techniques for sensor databases. In *International Conference on Data Engineering (ICDE)*.

CRISTESCU, R., BEFERULL-LOZANO, B., AND VETTERLI, M. 2004. On network correlated data gathering. In *IEEE INFOCOM*.

DALE, M. R., DIXON, P., FORTIN, M.-J., LEGENDRE, P., MYERS, D. E., AND ROSENBREG, M. S. 2002. Conceptual and mathematical relationships among methods for spatial analysis. In *Ecography 25*, 5.

DESHPANDE, A., GUESTRIN, C., MADDEN, S. R., HELLERSTEIN, J. M., AND HONG, W. 2004. Model-driven data acquisition in sensor networks. In *VLDB*.

FANG, Q., ZHAO, F., AND GUIBAS, L. 2003. Lightweight sensing and communication protocols for target enumeration and aggregation. In *MobiHoc*.

FERNANDEZ, C. AND GREEN, P. J. 2002. Modeling spatially correlated data via mixtures: a Bayesian approach. In *Royal Stat. Soc., Series B (Statistical Methodology) 64*, 802–826.

GANESAN, D., GREENSTEIN, B., PERELYUBSKIY, D., ESTRIN, D., AND HEIDEMANN, J. 2003. An evaluation of multiresolution storage for sensor networks. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

GIBBONS, P. B. 2001. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *VLDB*.

GOEL, A. AND ESTRIN, D. 2003. Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

GOEL, S. AND IMIELINSKI, T. 2001. Prediction-based monitoring in sensor networks: taking lessons from MPEG. In *ACM Comput. Comm. Rev. (CCR)*.

GUESTRIN, C., BODI, P., THIBAU, R., PASKI, M., AND MADDEN, S. 2004. Distributed regression: an efficient framework for modeling sensor network data. In *Information Processing in Sensor Networks (IPSN)*.

GUPTA, H., NAVDA, V., DAS, S. R., AND CHOWDHARY, V. 2005. Efficient gathering of correlated data in sensor networks. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*.

HALLEY, J. M., HARTLEY, S., KALLIMANIS, A. S., KUNIN, W. E., LENNON, J. J., AND SGARDELIS, S. P. 2004. Uses and abuses of fractal methodology in ecology. In *Ecology Letters*.

HEINZELMAN, W. R., CHANDRAKASAN, A., AND BALAKRISHNAN, H. 2000. Energy-efficient communication protocol for wireless microsensor networks. In *Hawaii International Conference on System Sciences (HICSS)*.

HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D., AND PISTER, K. 2000. System architecture directions for network sensors. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

HUSKER, A., KOHLER, M., AND DAVIS, P. 2003. Seismic amplitude variations due to site and basin edge effects in the Los Angeles Basin. In *Trans American Geophysical Union*.

INTANAGONWIWAT, C., GOVINDAN, R., AND ESTRIN, D. 2000. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *International Conference on Mobile Computing and Networking (Mobicom)*.

JAIN, A., CHANG, E. Y., AND WANG, Y.-F. 2004. Adaptive stream resource management using Kalman filters. In *ACM SIGMOD*.

JINDAL, A. AND PSOUNIS, K. 2004. Modeling spatially-correlated sensor network data. In *IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*.

LENNON, J. 2000. Red-shifts and red herrings in geographical ecology. In *Ecography 23*, 1.

LEVIS, P., LEE, N., WELSH, M., AND CULLER, D. 2003. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

MADDEN, S. R., FRANKLIN, M. J., HELLERSTEIN, J. M., AND HONG, W. 2002. TAG: Tiny AGgregation service for ad-hoc sensor networks. In *Symposium on Operating Systems Design and Implementation (OSDI)*.

MAINWARING, A., SZEWCZYK, R., POLASTRE, J., AND ANDERSON, J. 2005. Habitat monitoring on Great Duck Island. In http://www.greatduckisland.net.

MANJESHWAR, A. AND AGRAWAL, D. P. 2001. TEEN: A Routing protocol for enhanced efficiency in wireless sensor networks. In *International Workshop on Parallel and Distributed Computing, Issues in Wireless Networks and Mobile Computing (IPDPS)*.

MANJESHWAR, A. AND AGRAWAL, D. P. 2002. APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In *International Workshop on Parallel and Distributed Computing, Issues in Wireless Networks and Mobile Computing (IPDPS)*.

NATH, S., GIBBONS, P., ANDERSON, Z., AND SESHAN, S. 2004. Synopsis diffusion for robust aggregation in sensor networks. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

OLSTON, C., JIANG, J., AND WIDOM, J. 2003. Adaptive filters for continuous queries over distributed data streams. In *ACM SIGMOD*.

PATTEM, S., KRISHNAMACHARI, B., AND GOVINDAN, R. 2004. The impact of spatial correlation on routing with compression in wireless sensor networks. In *Information Processing in Sensor Networks (IPSN)*.

ROSSI, L. A., KRISHNAMACHARI, B., AND KUO, C.-C. J. 2004. Distributed parameter estimation for monitoring diffusion phenomena using physical models. In *IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*.

SCHABENBERGER, O. AND GOTWAY, C. A. 2005. Statistical methods for spatial data analysis. In *Chapman and Hall/CRC*.

SHARAF, M. A., BEAVER, J., LABRINIDIS, A., AND CHRYSANTHIS, P. K. 2003. TiNA: A scheme for temporal coherency-aware in-network aggregation. In *ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDe)*.

SZEWCZYK, R., MAINWARING, A., POLASTRE, J., AND CULLER, D. 2004. An analysis of a large scale habitat monitoring application. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

TOBLER, W. R. 1970. A computer movie simulating urban growth in the Detroit region. In *Economic Geography 46*, 2.

WOO, A., MADDEN, S. R., AND GOVINDAN, R. 2004. Networking support for query processing in sensor networks. In *Comm. ACM (CACM)*.

XU, N., RANGWALA, S., CHINTALAPUDI, K. K., GANESAN, D., BROAD, A., GOVINDAN, R., AND ESTRIN, D. 2004. A wireless sensor network for structural monitoring. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

YAO, Y. AND GEHRKE, J. 2003. Query processing for sensor networks. In *Biennial Conference on Innovative Data Systems Research (CIDR)*.

YOON, S. AND SHAHABI, C. 2005a. An experimental study of the effectiveness of Clustered AGgregation (CAG) leveraging spatial and temporal correlations in wireless sensor networks. In *USC Computer Science Department Tech. Rep. 05-869*.

YOON, S. AND SHAHABI, C. 2005b. Exploiting spatial correlation towards an energy efficient Clustered AGgregation technique (CAG). In *IEEE International Conference on Communications*.

ZHAO, J., GOVINDAN, R., AND ESTRIN, D. 2003. Computing aggregates for monitoring wireless sensor networks. In *IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*.

Received August 2005; revised February 2006, June 2006; accepted September 2006