Approximate Monitoring in Wireless Sensor Networks

Xingbo Yu, Sharad Mehrotra, Nalini Venkatasubramanian, Weiwen Yang
School of Information and Computer Science
University of California, Irvine, CA 92697, USA
{xyu, sharad, nalini, wyang}@ics.uci.edu

Abstract

In this paper, we develop an energy efficient approach to processing continuous aggregate queries in sensor networks with bounded quality constraints. Specifically, we exploit quality-aware in-network aggregation of query information to reduce communication costs. Given cluster structures over sensor nodes, we develop an adaptive approximate data processing protocol to aggregate data and process queries given error tolerance on user queries. The protocol enforces group quality constraints on clusters and adjusts error bounds to balance workload of clusters and reduce communication cost. It also tries to initiate local communication instead of global communication in the effort of minimizing overall communication overhead. Our experimental results indicate that significant benefits can be achieved by using our adaptive protocol.

1 Introduction

Advances in MEMS technology has enabled the development of small, inexpensive electronic microsensor devices (e.g. Berkeley Motes [11], MIT μamps nodes [3]) that integrate radio, sensing and processing components. This, along with the development of portable operating systems that can be embedded into sensor devices (e.g. TinyOS [8]) has made the deployment of "drop and play" applications a reality. These developments catalyze a wide spectrum of applications involving large scale sensor networks, such as military surveillance, biological monitoring and fault detection in civil infrastructures. Data in such monitoring based sensor networks is generated continuously, often at a very rapid rate.

A major constraint in the effective wide-scale deployment of such sensor networks is the limited battery power that the devices rely on for their functionality. For example, with no power management, the Mica motes [11] can last approximately 7-10 days. Since sensors are usually placed in environments with harsh conditions or embedded in remote rugged terrain (e.g. ecosystem monitoring), it is often difficult, if not impossible, to replenish or replace the batteries. Developing energy-efficient strategies for deploying wireless sensor network applications is of utmost importance. Due to the fact that communication cost in sensor networks is much higher than computation cost [20], substantial energy savings can be achieved by reducing the communication cost, sometimes even at the cost of more computation. Much of the prior work focuses on low duty-cycle operation for network related operations such as routing and media access with radio transmitters and receivers turned off whenever possible to save power [10, 9]. We consider, however, a technique that trades data quality for savings in communication cost, when approximate answers can be accepted by users. Another concern with wireless sensor network protocols is their ability to achieve uniform power consumption among the network components to extend the overall lifetime of the network without loss of connectivity. We show how the proposed technique can be exploited to achieve this goal by adjusting error tolerances for sensors and sensor groups.

Error is inherent in sensor networks. Variations in the sensing process, environmental factors and communication latencies introduce noise and imprecision into the sensor data received at the aggregation node. Although sophisticated techniques have been devised to improve the quality of sensor generated data, such as sensor fusion [2], exact results are difficult to achieve. This implies that even if sensor readings are captured exactly, it is still impossible to have a precise representation of real-world phenomena. Fortunately, in practice, monitoring queries usually require information about a group of sensors at a specific level of accuracy. This enables that the error constraints be defined on groups of sensors (not on an individual sensor). While the basic nature of the inaccuracy remains unchanged, the protocol designer now has a margin of flexibility - not every violation of accuracy on sensor needs to be reported to the server. In-network aggregation can specifically help eliminate messages to the server if the group quality constraints are satisfied even though sensor error bounds may be violated. This is especially useful under conditions of random noise where early aggregation can take advantage of the fact that positive and negative noise factors can cancel each other out.

Researchers are beginning to explore sensor database

technology, from semantics, architecture, to in-network query processing [16, 23, 17]. In network community, cluster-based routing and data gathering has been identified to be the most efficient communication technique for large-scale wireless sensor networks. Several protocols have been proposed [7, 15, 24]. Clusters are constructed to gather data locally before the data is forwarded to a server for final processing. The clusters can form a hierarchical structure. In this paper, we consider aggregate monitoring queries over a set of sensors in a wireless sensor network. A large class of sensor network applications are primarily designed to monitor the physical world; continuous queries are a natural fit in this scenario. The goal of this paper is to devise techniques and protocols for adapting the processing of aggregate monitoring queries in a sensor network in order to (a) satisfy monitoring quality constraints (b)minimize cost of query processing and (c)maximize the lifetime of the sensor network by balancing the workload at the sensor nodes. We present an adaptive protocol for data aggregation that minimizes communication overhead while guaranteeing error-bounds. The protocol also adaptively balances the workload of the clusters by manipulating the associated sensor values and the error tolerances of individual sensors.

The rest of the paper is organized as follows. In Section 2, we introduce some necessary background and formulate the approximate monitoring problem. Section 3 presents an adaptive approximate aggregate monitoring protocol(A^3) to process aggregate queries in a sensor network. Section 4 discusses the adaptive adjustment of error tolerances. Section 5 evaluates the performance of the proposed strategies for clustering and adaptation. We review related works in section 6.

2 The Adaptive Aggregation Problem

2.1 Sensor Networks

The sensor nodes in the networks we consider in this paper have capabilities similar to Mica Motes [11] which have a 4 MHz ATMEL processor, an embedded sensor, an A/D converter, and a RFM TR1000 radio with data rate at 40 kbits/second. The networks have a super node, an access point (AP) with unlimited resources (in terms of computation and communication) that interacts freely with remote users in a wide area network(where queries originate). All sensor nodes are scattered around the access points to form a connected network. The network topology is relatively stable. Communication between sensors and between a sensor and the AP occurs via multi-hop messaging. The sensor nodes are homogeneous and generate single attribute data. They are also clock-synchronized[1] which allows each newly generated data to be assigned a *round number*. A *round* is a processing cycle for the network to collect all data generated at a specific time instant. Sensor nodes may also have other features such as location-awareness.

The data to be aggregated is generated at a pre-known subset of sensor nodes, as specified in the query. Usually they are distributed in the geographical regions where the physical phenomena are being monitored. We assume these nodes are grouped into a set of clusters using cluster-oriented techniques such as those proposed in [7, 15, 24]. The clustering process is implicitly designed as part of routing algorithms. Data can be aggregated at cluster-heads before the aggregated results are reported to the access point. A cluster-head is physically represented by a sensor node. On the other hand, a sensor node can logically represent more than one cluster-head.

Sensor Data Representation: In this paper, we assume that the values captured by sensors are numerical. To support the notion of error tolerance, an approximation of the numerical sensor values is represented in the sensor database as an interval. Instead of using an upper- and lower-bound, we use the center value (v) and an error tolerance ($\delta, \delta > 0$) to represent the interval, which is $[v - \delta, v + \delta]$. Such a representation allows us to manipulate and adapt the value and error tolerance of a sensor node independently in our adaptive approximate aggregation protocol (Section 3). The data is also time-stamped with a logical value of time (a round number, in our case).

Cost Metrics: Due to the fact that communication is the dominant cost factor as compared to sensing and computing in normal data collection applications, we only model the communication cost. Communication cost can be measured in a number of ways - number of messages, total network load etc. We specifically evaluate architectures and protocols in terms of the *number of message-hops* generated in the network. One message-hop is one communication operation that transfers one packet of information between two neighboring sensors in one hop. Considering the fact that radio transmission ranges of sensor nodes, which decide power consumption, are set to be identical in common practice, the communication cost is directly proportional to the number of messages and the number of hops each message has to traverse. The significant advantage of the *total message hops* metric is that it indirectly covers other factors that can contribute to communication overhead in a sensor network, such as energy consumption for powering electronic sensor devices and overhearing cost[2]. As is the case with aggregation queries, we assume that all messages, including control messages are of the same size; in other words, an aggregate data packet will be of the same size as the data that has been aggregated within it. This approach is conservative since our protocol produces many control messages and they are usually smaller than sensor data messages.

[1] Clock synchronization is an important research issue [6] in wireless sensor networks. But it is beyond the scope of this paper.

[2] Cost for a sensor to catch packets and check the destinations of the packets[22].

2.2 Definitions and Objectives

We consider continuous **Approximate Aggregation Queries** $Q(S_1, S_2, ..., S_n, \delta)$ with an error bound δ on the result of the aggregate function over a set of sensors $S = \{S_1, S_2, ...S_n\}$. The error tolerance allows the query result to be approximate, hence the term *approximate aggregation query*. **Aggregation Tree(**AT**)** is an abstract tree structure constructed for approximate data aggregation. Leaf nodes in the tree represent sensor nodes and the intermediate tree nodes represent cluster-heads that reside on some sensor nodes. Each subtree corresponds to a cluster at a certain level. **Adaptive Approximate Aggregation Protocol (**A^3**)** is a protocol on top of AT that shows how aggregation can be performed to achieve energy-efficiency, as well as how to partially achieve uniform power consumption in sensor networks.

The problem of approximate aggregation can be formally stated as following: *Given a wireless sensor network with an abstract cluster structure and a continuous approximate aggregation query* $Q(S_1, S_2, ..., S_n, \delta)$ *over a set of sensors* $\{S_1, S_2, ...S_n\}$ *in the network with a specific quality requirement, design a data aggregation protocol that can answer the continuous query successfully, while allowing the sensor network achieve maximum lifetime.* Thus, the ultimate goal of our techniques is to maximize lifetime of sensor networks while processing user queries successfully. Two immediate subgoals are reducing communication cost and balancing the workloads within sensor networks. We achieve these goals through the following steps:

- Design an A^3 protocol to minimize communication cost while preserving quality in answers to user queries.
- Enhance A^3 protocol to adjust error tolerances of clusters at different levels adaptively so that workload among clusters are balanced.

Note that the logical AT structure and protocol proposed in this work are devised to facilitate processing of general aggregate monitoring queries (the focus of this paper). We use a summation query to illustrate our protocol in the remainder of this paper. Extensions to other aggregate queries, i.e. min/max are discussed in Appendix A. We present our proposed techniques to address the above issues in the next two sections.

3 Data Collection via Adaptive Approximate Aggregation

Given a cluster structure of a sensor network, precise data collection through aggregation is straightforward. In this section, we will show that whenever approximate results are acceptable, extra savings can be achieved by developing an adaptive approximate data aggregation (A^3) protocol.

3.1 Properties of Hierarchical Cluster Architectures

Cluster structures can be hierarchical when cluster-heads are clustered. This corresponds to multi-level communication strategy when, e.g., sensor nodes talk to super nodes and super nodes talk to access points, both through wireless communication; then access points communicate with a server through wired links. Figure 1 shows an aggregation tree (AT) for a 2-level cluster architecture. In general, each branch of the AT may have different depth and that leaf nodes are source sensor nodes.

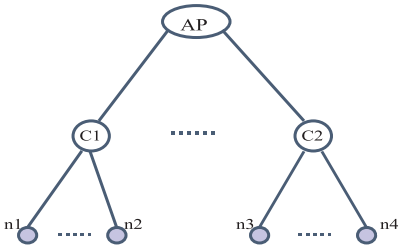

Figure 1. 2-Level Cluster Architecture

With a tree structure, a precise data collection process works as follows. Each sensor node (a leaf node in the tree) sends data (captured values) to its parent node. As a special case, a node which also serves as a cluster-head will have its value available to the cluster-head without communication cost. The parent node will report an aggregate (over all child nodes) value upwards after it hears from all children. The top level cluster-head(an AP) will produce a value aggregated over all relevant sensor nodes at the last step of the round. The presence of synchronized clocks ensures that data at different rounds can be distinguished.

When a predetermined error in the aggregation result can be tolerated, we observe that it is not necessary for a node to report every new value (new sensor reading or new aggregated value) if this new value is very close to a default value that its parent is aware of. To be more specific, if a parent node keeps in its cache a default value v and a default error bound δ for each child node, each node reports a new value to its parent node only when the error bound is violated by the new value v' ($|v' - v| > \delta$). When a parent node does not hear from a child node, it assumes the child node has a value within $[v-\delta, v+\delta]$. In the more general case, the new value could be compared with a "predicted" value of the node to check violation, if a prediction model is implemented at the node and its parent. A prediction model[14] is a model that two nodes agree on and is used to generate a new value at the parent if the child node does not report an updated value. In this paper, we discuss a process by which a parent node assigns to each child node an artificial future value (using the previously reported value is one of many possible choices). We refer to this artificial future value as the *default value*. The setting of default values must be such

that accuracy constraints of the final aggregation result are satisfied. In order to maintain correctness of the final result, some invariances have to be maintained on the default values and error bounds. More specifically, we have the following theorems.

Theorem 3.1 *Let v_i and δ_i represent default values and error tolerances of member nodes in a cluster, v_c and δ_c represent the aggregate value and error tolerance at cluster-head, and N_c be the total number of member nodes of the cluster. The following inequality is a necessary condition for a SUM aggregation to be evaluated correctly: $\delta_c \geq \sum_{i=1}^{N_c} \delta_i$.*

Proof: To maintain answer correctness on SUM aggregation, range $[\sum_{i=1}^{N_c} v_i - \sum_{i=1}^{N_c} \delta_i, \sum_{i=1}^{N_c} v_i + \sum_{i=1}^{N_c} \delta_i]$ has to be included in $[v_c - \delta_c, v_c + \delta_c]$. This implies $\sum_{i=1}^{N_c} v_i - \sum_{i=1}^{N_c} \delta_i \geq v - \delta$ and $\sum_{i=1}^{N_c} v_i + \sum_{i=1}^{N_c} \delta_i \leq v + \delta$. They can be further simplified as $\sum_{i=1}^{N_c} v_i - v_c \geq \sum_{i=1}^{N_c} \delta_i - \delta_c$ and $\sum_{i=1}^{N_c} v_i - v_c \leq \delta_c - \sum_{i=1}^{N_c} \delta_i$, which is the same as $\delta_c \geq \sum_{i=1}^{N_c} \delta_i$. ■

Theorem 3.2 *: $v_c = \sum_{i=1}^{N_c} v_i$ and $\delta_c \geq \sum_{i=1}^{N_c} \delta_i$ are sufficient conditions to maintain answer correctness for SUM aggregation.*

Proof: With the first condition, the default range for summation $[\sum_{i=1}^{N_c} v_i - \sum_{i=1}^{N_c} \delta_i, \sum_{i=1}^{N_c} v_i + \sum_{i=1}^{N_c} \delta_i]$ is equivalent to $[v_c - \sum_{i=1}^{N_c} \delta_i, v_c + \sum_{i=1}^{N_c} \delta_i]$. Using the second condition, we conclude $v_c - \sum_{i=1}^{N_c} \delta_i \geq v_c - \delta_c$ and $v_c + \sum_{i=1}^{N_c} \delta_i \leq v_c + \delta_c$. This implies the summation value will be bounded by $[v_c - \delta_c, v_c + \delta_c]$. ■

In addition to maintaining the answer correctness of aggregation queries, large error tolerances on cluster member nodes are desired in order to reduce communication cost. In fact, we want to have the maximum possible total error tolerance for cluster members. The following corollary gives us such conditions.

Corollary 1 *The necessary and sufficient condition for SUM aggregation to be evaluated correctly and for cluster members to have maximum total error tolerance is:*

$$v_c = \sum_{i=1}^{N_c} v_i, \quad (1)$$

$$\delta_c = \sum_{i=1}^{N_c} \delta_i. \quad (2)$$

Proof: Sufficiency immediately follows theorem 4.2. To show necessity, note that maximum total error tolerance implies $\delta_c = \sum_{i=1}^{N_c} \delta_i$. This in turn requires $v_c = \sum_{i=1}^{N_c} v_i$ to maintain aggregation correctness. ■

Equations (1) and (2) give us two invariances we want to maintain for SUM queries. For other type of aggregation functions, the invariances are similar. See Appendix A for detailed discussion.

The two invariances postulated above allow for safe manipulation of the default values and error bounds. To efficiently deploy the hierarchical cluster structure for approximate data aggregation, we adaptively set default values and adjust error bounds based on the level of change at each sensor node while ensuring that the invariances be maintained. We illustrate how the updates are performed using a 2-level cluster structure and then generalize the protocol to multi-level cluster structures.

3.2 Adaptive Data Updates

We illustrate how to adapt default values for a 2-level cluster structure as the one shown in figure 1. We consider the monitoring of summation of a set of sensor readings. With the SUM query, one simple way to distribute overall error tolerance among all nodes in the tree is to evenly distribute tolerance of a parent node to each child node. (An alternative, which we are not considering, could be to keep some slack for a parent node and distribute the other part to children. We haven't found in our experiments any evidence to show that this method would reduce the number of necessary messages.)

In figure 1, a new data value (e.g. $v_1^{'}$) is generated at a leaf node (source sensor, e.g. n_1). n_1 will first check the value against its default value (v_1) and error bound (δ_1). If $|v_1^{'} - v_1| \leq \delta_1$, nothing needs to be done. The lack of a message from n_1 will tell the parent node (c_1) that the value of n_1 in this round is within the interval $[v_1 - \delta_1, v_1 + \delta_1]$. When a violation of error bounds at the sensor node occurs, there are 2 cases that must be considered separately. In the first case, one or more child nodes violate their error bounds, and the bound on parent node is also violated. (As a special case, if only one child has its bound violated, the bound of parent node must be violated.) The bound on a parent node is violated when

$$\sum_{i=1}^{N_v} (v_i^{'} - v_i) > \sum_{i=1}^{N_v} \delta_i, \quad (3)$$

where N_v is the total number of child nodes that have violated their bounds and reported to their cluster-head. The implicit assumption that has to be made here is that those children who didn't violate their bounds may be very close to violating the bounds in the worst case.

Assume n_1 and n_2 violate their bounds and report new values $v_1^{'}$ and $v_2^{'}$ to c_1. Then n_1 and n_2 will reset their default values to $v_1^{'}$ and $v_2^{'}$ after waiting for a predefined time period without hearing from c_1. (This delay is important for maintaining cache consistency with c_1 due to the presence of the second case described below.) c_1 will also report its new value to its parent (the AP in our example), since with the worst case assumption on other non-reporting child nodes, the error bound on c_1 will be violated.[3] The new value in this case is not a single value,

[3] An alternative is to probe other child node for their exact values. We are not taking this approach.

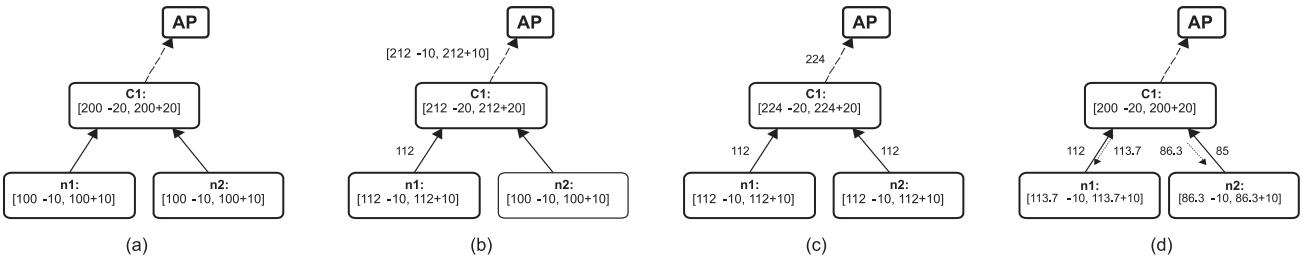

Figure 2. An Example of Data Updating in A^3 Protocol

but an interval $[v_{c1} + (v_1^{'} - v_1) + (v_2^{'} - v_2) - (\delta_{c1} - \delta_1 - \delta_2), v_{c1} + (v_1^{'} - v_1) + (v_2^{'} - v_2) + (\delta_{c1} - \delta_1 - \delta_2)]$. As a result of updates from child nodes, the width of the range is smaller than that given by the error bound at c_1, which indicates that c_1 updates a more accurate value interval to the AP. After reporting the new value, c_1 resets its default value to $v_{c1} + (v_1^{'} - v_1) + (v_2^{'} - v_2)$ and resets its cache values for n_1 and n_2 to $v_1 = v_1^{'}$ and $v_2 = v_2^{'}$.

The second case is that more than one child node reports new values to c_1, but the bound on c_1 is not violated. In this case, c_1 will not report to its parent node because the group quality constraint is still satisfied. However, since invariance (1) may be violated between c_1 and its child nodes, reported values from child nodes *can not* serve as default values for future operations directly(without adjusting error tolerances). Thus, c_1 needs to compute desired new default values for each child who reported in the round and inform them to reset their default values. When computing the new default values, we try to keep the new values close to the updated values from children. This can be achieved by solving the following equations.

$$\sum_{i=1}^{N_v} v_i = v_{c1} - \sum_{j=N_v+1}^{N_{c1}} v_j \quad (4)$$

$$\frac{v_i}{v_i^{'}} = CONST \quad (5)$$

Here we assume the first N_v ($N_v < N_{c1}$) child nodes reported their new values. $CONST$ is a constant and v_i is the desired new default value for the i^{th} ($1 \leq i \leq N_v$) child node. Solving the equations, we get

$$v_i = \frac{v_i^{'}}{\sum_{k=1}^{N_v}(v_k^{'})}(v_{c1} - \sum_{j=N_v+1}^{N_{c1}}(v_j)) \quad (6)$$

The fact that c_1 does not report to the AP in this case yields some savings on communication cost. Although additional computation and control messages within the cluster may be initiated, they are usually not as expensive since computation and local communication within clusters are cheaper. Our protocol also benefits from the fact that there is one child node whose value is always available, which is the sensor node that the cluster-head (AT node) resides on. There is no communication cost between these two nodes. So we always have the information from the sensor node available to help make better decisions on whether to report or not.

A simple example, as shown in figure 2, illustrates the updating process. Assume that c_1 has only two children n_1 and n_2, and we have the following values initially: $v_{c1} = 200, \delta_{c1} = 20, v_1 = 100, \delta_1 = 10, v_2 = 100, \delta_2 = 10$ (see figure 2(a)). Suppose now error bound violation occurs at node n_1 with $v_1^{'} = 112$, and there is no report from node n_2. n_1 will set its default value to $v_1 = 112$. c_1 will set its default value to $v_{c1} = 212$ and report $[212-10, 212+10]$ to the AP (figure 2(b)). Note that we haven't discussed adjustment of error tolerance. The error bound is still 20. So the default interval for c_1 is now $[212 - 20, 212 + 20]$. If in the same round, the bound on n_2 is also violated by a new value $v_2^{'} = 112$, v_2 will also be set to 112. And c_1 will report 224 to the AP and reset its default value to $v_{c1} = 224$ and default interval to $[224 - 20, 224 + 20]$ (see figure 2(c)).

Let us consider another case where in the same round, the bound on n_2 is also violated, but now by a new value $v_2^{'} = 85$, the bound for c_1 is not violated since $200-20 < 112 + 85 < 200 + 20$. Thus c_1 will not propagate an update to the AP, neither will its default value be changed. However, the default values for n_1 and n_2 will have to be re-evaluated by c_1 using formula 6. The results, $v_1 = 113.7$ and $v_2 = 86.3$, will be sent to the corresponding child node.

In our 2-level architecture, there are no further actions after resetting of default values since n_1 and n_2 are at the lowest level of the AT tree. However, in the case of a multi-level tree, there may be other levels of nodes below n_1 and n_2, the default values at those levels also need to be reset so that the effect can be "propagated" to all descendants of c_1. This, however should be performed only when reporting a new value from c_1 to AP is more expensive than the cost of resetting descendants of c_1.

4 Adaptive Adjustment of Error Tolerance

We have shown so far how default values can be maintained in the data updating process to ensure invariance (1). We also fixed the error bounds for each node so

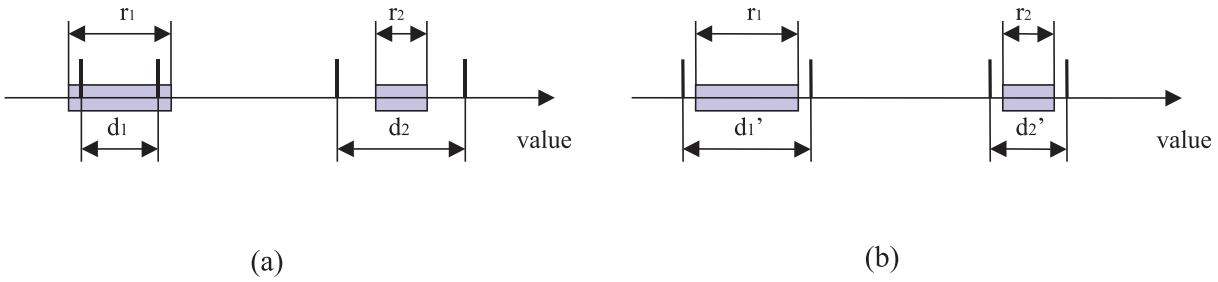

Figure 3. Distribute Error Tolerance Between Two Nodes

that invariance (2) also holds. But this may not be cost-efficient, since different source sensors may exhibit different types of behavior. For those sensors whose values do not change very much, it is beneficial to assign them smaller error tolerances and assign the slack saved this way to other more active sensor nodes. We exploit this insight to design an approximate algorithm to adaptively adjust the error tolerance at each node in order to achieve an overall balance on communication activity by allowing sensor nodes update at similar frequencies.

On the other hand, balanced workloads may also result in reduced overall workload, although optimality is not guaranteed due to the unknown nature of data behavior. Figure 3 shows one case that can illustrate the benefit of having balanced workload. Assume data readings of the two sensor nodes in different rounds are within some ranges (r_1 and r_2). And assume the total error tolerance allowed is also the same ($d_1 + d_2 = d_1' + d_2'$). It is obvious that the distribution in figure 3(a) will generate some messages while the distribution in figure 3(b) will not.

4.1 Adaptive Adjustment Technique

The best adjustment technique could be developed if the data behaviors can be predicted, which is impossible in real applications. Learning the distributions of data by fitting statistical models can also be a good solution. But due to the limited storage and computing power of sensor nodes, those strategies will be too complex to implement. We propose a simple strategy for estimating sensor node workloads based on update frequencies. Other factor such as resource constraints (e.g. power level) can be assimilated into the estimation as well. In the A^3 protocol, an adjustment is always initiated by a parent node who keeps in its cache default values and error bounds of all its children. In order to know the workloads of child nodes, the parent node also keeps a record of the update frequency, f, for all its children over certain period of time in history. This time period can be envisioned as a time-driven or round-driven sliding window. Then f is the number of updates a child node makes during the time window. In our experiments, we set all window sizes to 32 (rounds).

At each time an adjustment is activated at a parent node, our algorithm adjusts only two child nodes who have maximum and minimum update frequencies. An overall balance among sibling nodes will be achieved in a long run. We use the following criterion to initiate the adjustment:

$$\frac{f_{max} - f_{min}}{f_{max}} \geq r_{th}, \tag{7}$$

where f_{max} is the maximum frequency and f_{min} is the minimum. r_{th} is a threshold we choose for the algorithm. We represent error tolerances corresponding to f_{max} and f_{min} by δ_{fmax} and δ_{fmin}. Then we can reset the error tolerances using the following formulae:

$$\delta_{fmax} = \alpha\delta_{fmax} + (1-\alpha)(\delta_{fmax} + \delta_{fmin})\frac{f_{max}}{f_{max} + f_{min}} \tag{8}$$

$$\delta_{fmin} = \alpha\delta_{fmin} + (1-\alpha)(\delta_{fmax} + \delta_{fmin})\frac{f_{min}}{f_{max} + f_{min}} \tag{9}$$

In the formulae, α is an adjustment factor which allows us to keep part of the previous error tolerance and redistribute the other part between the two (min and max update frequency) nodes. The value of α should be chosen based on the sensitivity of the update value. Care should be taken to ensure that the choice of α do not cause any oscillation effect in which frequency of node with f_{max} becomes too small after the adjustment and that of node with f_{min} gets too large. The basic A^3 protocol can be enhanced by selecting r_{th} and α adaptively. Once an error adjustment action is taken by a tree node, the node waits for a fixed number of rounds before attempting another adjustment. This is necessary to factor in the time lag between the error adjustment and the actual impact on the update frequencies.

For a 2-level architecture, the AP and cluster-heads are in charge of the error bound adjustment process. For a multi-level structure, similar to the resetting process of default values, the effect of error bounds being changed must be propagated to descendants. Special care needs to be taken in the propagation since reducing the error bound on a node is an unsafe operation and may cause the total error bound on a node's children to be greater than the reduced bound on a node. This may result in incorrect results for monitoring queries. To handle this, we can introduce the notion of "safe operation" and "unsafe operation" similar to those used in [1]. An unsafe

operation can only be committed after the unsafe operations on child nodes are committed. In another words, if a node has its error bound reduced, it can not perform approximate aggregation until all children (and thus, descendants) have reduced their error bounds. On the other hand, enlarging error bound is a safe operation that will not yield incorrect results. The operation can be committed any time.

Procedure A^3 Protocol
INPUT:
v_i, $v_i^{'}$: default and new values of the i^{th} child;
δ_i : error bound of the i^{th} child; C : cluster;
N_c : total members in C; N_v : members reporting;
f_{max}, f_{min}: max and min update frequencies;
r_{th}, α : adjustment threshold and factor;
// Data Updating
(1) **while**(receive $v_i^{'}$ from child)
(2) **if**($\sum_{i=1}^{N_v}(v_i^{'} - v_i) > \sum_{i=1}^{N_v}\delta_i$)
(3) $v_c = v_c + \sum_{i=1}^{N_v}(v_i^{'} - v_i)$;
(4) update (v_c, $\delta_c - \sum_{i=1}^{N_v}\delta_i$) to parent;
(5) **endIf**;
(6) **else**
(7) $v_{inew} \leftarrow \frac{v_i^{'}}{\sum_{k=1}^{N_v}(v_k^{'})}(v_c - \sum_{j=N_v+1}^{N_c} v_j)$;
(8) reset v_{inew} as i^{th} child's default value;
(9) **endElse**;
(10) **endWhile**;
(11) **if**(recieve v_{inew} from parent)
(12) reset default values for all children;
(13) $v_i \leftarrow v_{inew}$;
(14) **endIf**;
// Error Bound Adjustment
(15) **if**(this is not leaf node and $\frac{f_{max}-f_{min}}{f_{max}} \geq r_{th}$)
(16) $\delta_{fmax} \leftarrow \alpha\delta_{fmax} + (1-\alpha)(\delta_{fmax} + \delta_{fmin})\frac{f_{max}}{f_{max}+f_{min}}$;
(17) $\delta_{fmin} \leftarrow \alpha\delta_{fmin} + (1-\alpha)(\delta_{fmax} + \delta_{fmin})\frac{f_{min}}{f_{max}+f_{min}}$;
(18) update δ_{fmax};
(19) update δ_{fmin};
(20) **endIf**;
(21) **if**(recieve δ_{new} from parent)
(22) **if**(this is not a leaf node)
(23) reset δ for all children;
(24) **endIf**;
(25) $\delta \leftarrow \delta_{new}$;
(26) **endIf**;

Figure 4. A^3 Protocol

Propagation can also be conducted based on the update frequencies. This process will also try to balance sensor node workloads using the least possible messages. Let n_s be the number of nodes whose update frequency is below average. We can choose the first $\frac{1}{2}n_s$ children with lowest update frequencies, and reduce their bounds evenly. For example, as shown in figure 2(a), suppose error bound for c_1 is reduced from ± 20 to ± 15 by the AP. If $f_1 < f_2$, δ_1 will be changed from ± 10 to ± 5.

We summarize the A^3 protocol in figure 4. Note that since a physical sensor node can host multiple logical nodes in cluster architecture, multiple copies of the protocol can run at the same physical sensor node.

4.2 Discussion

Several choices exist in the process of designing a sensor network protocol. The AT structure and A^3 protocol are specialized to address general monitoring tasks. Specific knowledge on the physical phenomena being monitored can enable further optimizations. For example, in monitoring a highly dynamic phenomenon, we may want to instantiate r_{th} to be a high value. Otherwise the resetting of error tolerance may happen too often for adjustment of the error bounds to be advantageous. Another design parameter to be determined is the window size for computing update frequency. A possible approach would be to choose a larger window size for nodes in higher levels allowing these nodes to make adjustment decisions more conservatively.

In our work, the estimation of workload does not consider energy levels of the individual sensor nodes. Further optimizations to the A^3 protocol can be achieved by integrating energy consumption models for sensor nodes into the adaptation process.

The cluster architecture itself may be further exploited to improve the performance of in-network processing. For instance, transmission schedules may be established that dictate exactly when individual cluster members can communicate in order to avoid collisions. Similarly, if packet routing functionality is limited to cluster-heads, other cluster members may be scheduled to sleep when they are not transmitting and receiving, further optimizing energy consumption. Implementing the above techniques requires closer integration with network and MAC layer protocols; this is a topic for future research.

5 Performance Evaluation

5.1 Experimental Setup

In the preliminary experimental study, we try to simulate a real sensor network application scenario. The experimental topology consists of a connected random graph on a 50×50 two-dimensional space. 120 sensor nodes are placed uniformly randomly on the space. We choose a radio coverage limit for all sensor nodes so that all neighbors within the range can be connected by an edge. This range is chosen so that the network is connected and the edges are not too dense. We then select 36 nodes from three corner areas as the relevant nodes for a sample query. Choosing this topology over a grid-based topology enables us capture the randomness of sensor placement which is often governed by factors other than geographic information. Cluster-heads in the three groups are chosen by using a modified version of small-size weakly-connected dominating set algorithm (see [4]) so that the total distance to the heads in each group is

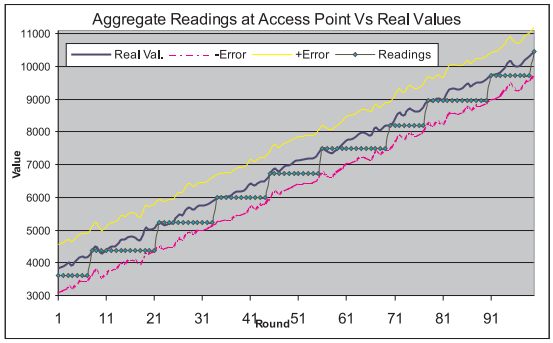

Figure 5. Aggregated Result at Access Point

minimal. The lengths (number of hops in shortest path) of the edges within a cluster vary between 1 and 3. Distances from the selected cluster-heads to the AP are 6, 14, and 9 hops.

Note that this type of randomly generated network is a conservative choice for testing A^3 protocol. When a network exhibits certain patterns, such as dense implementation in some areas, cluster structure will allow the A^3 protocol to perform more efficiently. An immediate observation after having an AT is that precise data aggregation using AT is much cheaper than a direct end-to-end approach for each round (399 vs 81 message-hops) .

Sensor Data: In the experiments, we simulate monitoring of a set of sensors that measure some physical world values (e.g. radiation level) that increase linearly. We model the sensor measurements as uniformly distributed random values within interval $[v - \rho \cdot \delta, v + \rho \cdot \delta]$. v is the ideal value for time instance obtained from a linear function. We initialize $\delta = \{0, 5, 10\}$ for each source sensor and $\rho = 1.2$ in our experimental setup, these factors are further manipulated in specific experiments.

5.2 Evaluating the A^3 Protocol

Update Cost with Fixed Error Tolerances: We execute the update protocol while fixing the error bounds which are initially evenly distributed among clusters and sensors. The aggregated result as seen at the AP is shown in figure 5 in which it is compared to the real aggregate that would result from precise data aggregation. We can see that the value being observed is within the specified error bound of the real result. An accuracy boost is observed periodically due to the violation of lower error bound, which is decided by the fact that the approximate aggregate tends to be smaller than the real aggregate since sensor values are generally increasing. Figure 6 shows (after a warm-up period) the update costs for each round with different error tolerances. We can see that the larger the error tolerance is, the more the savings. In our data setup, we achieve around 50% savings on communication cost with an absolute error tolerance of 10 for each sensor. We also observe that the cost in each round exhibits a pattern, with a few low costs followed by a high cost. This is due to the fact the protocol tends to cost less for a couple of rounds when the error bounds on cluster-heads are not violated. But when the local adjustment of default values can not catch up with the increase of the aggregated sensor values, a long range communication to the AP is activated, which in turn results in a peak in communication cost.

In addition to linearly increasing data, we also studied update cost for constant data with random noise. The results (which are not shown due to space limit) confirm our early observation that under random noise scenario, noise values tend to cancel each other out and the cost of updating is fairly low.

Performance of Error Tolerance Adjustment: We simulate dynamic behavior in sensor values by setting different intervals ($\rho = 1.2, 1.8, 2.4$) when choosing measurements for sensors in the three clusters. This cause measurements of some clusters to be very active ($\rho = 2.4$), and some to be less active ($\rho = 1.2$). We first choose different r_{th} and α values used in the error adjustment. The resulting cost for each round is shown in figure 7. Our cost evaluation is conservative since we assume control packets, which usually are smaller, are of the same size as data packets. Figure 8, 9, 10, and 11 show the update frequencies and cluster error tolerances of each cluster-head(the adjustment is activated at round 100). Note that in the figures, the three frequency (f_1, f_2, f_3) series correspond to the three clusters (c_1, c_2, c_3).

As shown by the figures, the adjustment causes error tolerances to deviate from the initial uniform setting, with larger error bounds assigned to cluster with more active measurement values. And as a result, the update frequencies get closer to each other in the later rounds of aggregation. When the error tolerance is non-zero, the overall cost in each round as shown in figure 7 is significantly reduced as compared with figure 6. We can also see the effect of parameter values on the performance of error adjustment. Figure 8 and 9 show that the frequencies are adjusted faster with a smaller α value. Figures 10 and 11 show that smaller r_{th} value causes error bound adjustment to be more sensitive to fluctuations in data values.

6 Related Work

Research areas related to our work include sensor databases, network aggregation, and quality-aware data collection. Recent works in sensor databases, such as the TinyDB project [16] and COUGAR project [23] investigate various issues in developing a sensor database. Madden et al. at UC Berkeley also investigated the implementation of aggregation as part of the system service architecture[17]. The work on quality aware sensor data management that we are doing in our QUASAR project [13] tries to trade data quality for network performance

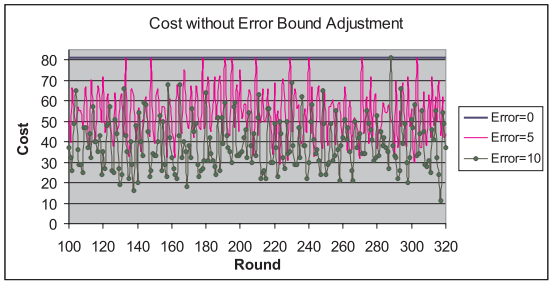

Figure 6. Cost for Data Updating

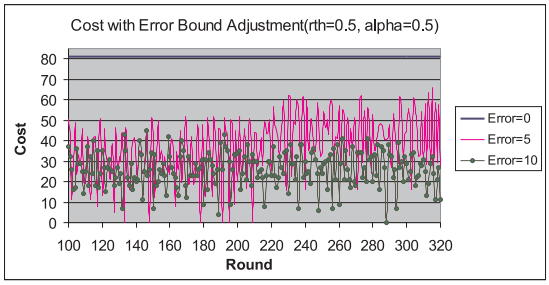

Figure 7. Cost with δ Adjusted

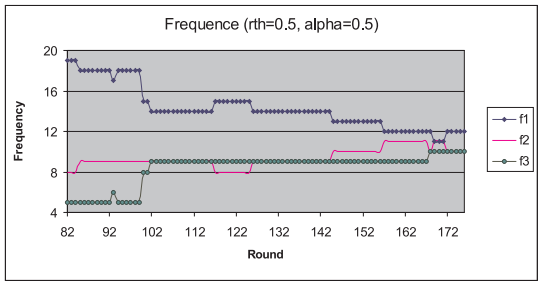

Figure 8. Update Frequency ($r_{th} = 0.5$, $\alpha = 0.5$)

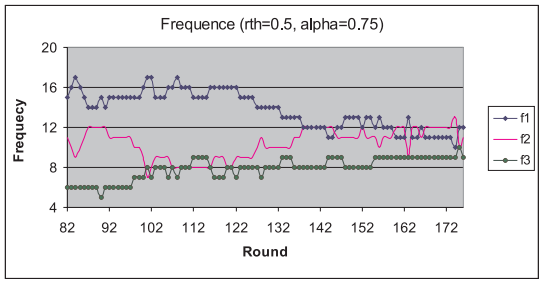

Figure 9. Update Frequency ($r_{th} : 0.5$, $\alpha : 0.75$)

in an efficient way.

In sensor network data gathering area, several approximate network clustering algorithms using weakly-connected dominating sets were presented in [4]. [12] and [5] target uniform energy dissipation to achieve maximum lifetime for a sensor network. LEACH [7] and PEGASIS [15] explore cluster-based data gathering protocols that rotate the cluster-head randomly to evenly distribute workload among sensor nodes. Directed Diffusion [10] provides a data dissemination approach in which a node requests data by propagating interests, in which aggregation is implemented through "path sharing". Aside from aggregation, power-aware routing [21] techniques have been recently developed to minimize the cost of intermediate routing nodes in ad-hoc networks.

Research in quality-aware data collection addresses the relation between quality requirements of query answers and quality of the raw data, with the ultimate goal of satisfying query requirements while minimizing certain cost. For example, Lazaridis et al.[14] developed an approximate data compression algorithm for data archiving. Our previous work [25] interprets quality of spatial range query results using probabilistic uncertainty. Olston et al.[19] provided an adaptive data collection protocol to collect data with certain precision requirements so that the application quality requirements of continuous queries can be met and the total communication cost for collecting the data is minimized. Constrained data caching [18, 1] deals with how to ensure consistency constraints dynamically in distributed databases. Our error bound adjustment method bears similarity to the works in this area to some extent.

Most of the previous works have ignored the underlying physical network that carries the data collection messages. Our approach takes this into consideration and exploits the hierarchical structure of networks to enforce group quality for each cluster. To the extent of authors' knowledge, we are not aware of any previous work on approximate data aggregation based on a hierarchical structure which enforces group quality constraints. Given the importance of sensor monitoring applications, we believe this is a critical issue that can impact performance in a variety of sensor application domains.

References

[1] D. Barbara and H. Garcia-Molina. The demarcation protocol: A technique for maintaining constraints in distributed database systems. *VLDB Journal*, 2(3), 1994.

[2] R. Brooks and S. Iyengar. *Multi-Sensor Fusion: Fundamentals and Applications with Software*. Prentice Hall, 1998.

[3] A. Chandrakasan, R. Min, M. Bhardwaj, S. Cho, and A. Wang. Power aware wireless microsensor systems. In *ESSCIRC*, 2002.

[4] Y. Chen and A. Liestman. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In *MobiHoc*, 2002.

[5] K. Dasgupta, K. Kalpakis, and P. Namjoshi. An efficient clustering-based heuristic for data gathering and aggregation in sensor networks. In *WCNC*, 2003.

[6] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *IPDPS Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, 2001.

[7] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS*, 2000.

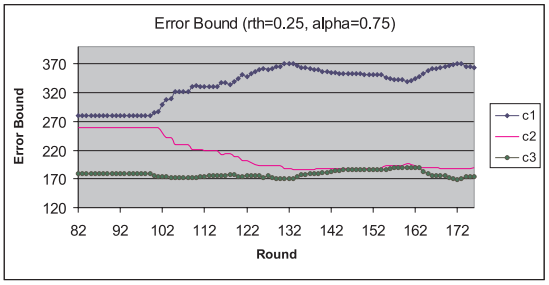

Figure 10. Error Bounds (r_{th} : 0.25, α : 0.75)

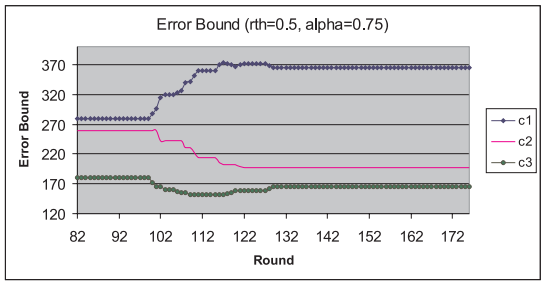

Figure 11. Error Bounds (r_{th} : 0.5, α : 0.75)

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

[9] C. J. Hughes, J. Srinivasan, and S. V. Adve. Saving energy with architectural and frequency adaptations for multimedia applications. In *MICRO*, 2001.

[10] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *MobiCOM*, 2000.

[11] J. Kahn, R. Katz, and K. Pister. Next century challenges: Mobile networking for 'smart dust'. In *MOBICOM*, 1999.

[12] K. Kalpakis, K. Dasgupta, and P. Namjoshi. Maximum lifetime data gathering and aggregation in wireless sensor networks. In *NETWORKS*, 2002.

[13] I. Lazaridis, Q. Han, X. Yu, S. Mehrotra, N. Venkatasubramanian, D. V. Kalashnikov, and W. Yang. Quasar: Quality-aware sensing architecture. In *SIGMOD Record*, 2004(to appear).

[14] I. Lazaridis and S. Mehrotra. Capturing sensor-generated time series with quality guarantees. In *ICDE*, 2003.

[15] S. Lindsey and C. S. Raghavendra. Pegasis: Power efficient gathering in sensor information systems. In *Proceedings of IEEE Aerospace Conference*, 2002.

[16] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD*, 2003.

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

[18] S. Mazumdar, M. Pietrzyk, and P. K. Chrysanthis. Caching constrained mobile data. In *CIKM*, 2001.

[19] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD*, 2003.

[20] G. Pottie and W. Kaiser. Wireless sensor networks. *Communications of the ACM*, 43, 2002.

[21] S. Singh and C. S. Raghavendra. Pamas: Power aware multi-access protocol with signalling for ad hoc networks. *ACM Computer Communications Review*, 1999.

[22] X. Yang and A. Bouguettaya. Broadcast-based data access in wireless environments. In *EDBT*, 2002.

[23] Y. Yao and J. Gehrke. Query processing in sensor networks. In *CIDR*, 2003.

[24] F. Ye, G. Zhong, S. Lu, and L. Zhang. Gradient broadcast: A robust data delivery protocol for large scale sensor networks. *WINET*, 11(2), March 2005.

[25] X. Yu and S. Mehrotra. Capturing uncertainty in spatial queries over imprecise data. In *DEXA*, 2003.

A Approximate *MAX/MIN* Monitoring

As discussed in section 3, in approximate aggregate monitoring, two invariances need to be maintained to ensure correctness of the final result. Our protocol has been illustrated using the invariance for SUM query as shown in equation 1. The Average (AVG) query has similar invariance and shares a similar protocol. The invariances for *MAX/MIN* queries are different and in fact less complicated. For *MAX* queries, the invariances are

$$v_c = max\{v_i\}, \quad (10)$$

$$\delta_c = max\{\delta_i\}. \quad (11)$$

Invariance 11 indicates that we should set error tolerances for all child nodes at a same value. Since cluster-head will take the maximum, it is only advantageous to set all nodes to the maximum error bound. This further implies that all nodes in the AT share the same error bound. With identical error bound, the adjustment component of the protocol disappears. A cluster-head only needs to monitor the maximum default value among its child nodes. To do this, a cluster-head keeps in its cache all default values of its children. When it receives new values from some children in a specific round (which implies error bound violations occur with these nodes), it resets the corresponding default values in its cache. Next, if the maximum value among all the cached default values is changed from the previous round, the new maximum value is sent to its parent node. Note the protocol is simpler since, besides identical error bounds, there is no need for a parent node to reset default values of its children. The default values are automatically set at their reported values. *MIN* aggregates can be monitored with exactly the same protocol.