Energy-Conscious Data Aggregation Over Large-Scale Sensor Networks

Fatih Emekci Hailing Yu Divyakant Agrawal Amr El Abbadi

Abstract

Recent advances in hardware technology facilitate applications requiring large numbers of sensor devices, where each sensor device has computational, storage, and communication capabilities. Since sensor devices are powered by ordinary batteries, power is a limiting resource in sensor networks. Power usage can be reduced by pushing part of the computation into the network to reduce communication cost, which is the main energy consumer in sensor networks. In order to further reduce power usage, we propose power-aware query processing techniques for aggregation queries. Instead of requiring exact answers to queries, we introduce precision into queries to give users full control of the tradeoffs between precision and energy usage. Our query processing approach incorporates in-network prediction to further reduce the need for constant communication. Moreover, we optimize the execution of multiple queries to take advantage of sharing common aggregated values among different queries. Since communication is three orders of magnitude more expensive than computation, incorporating precision and efficiently executing multiple queries results in significant power savings, thus extending the lifetime of sensor networks.

1 Introduction

Due to advances in miniaturization, low power, and low cost design of sensors, large-scale sensor networks are becoming a reality and are being used in many applications, such as environment monitoring on Great Duck Island and James Reserve [1, 10], where sensor nodes collect light, temperature, and humidity data. In such networks, each sensor node has the capabilities of sensing, computing, and communicating. For example, one of the current commercially available sensor devices, Berkeley MICA [4], has the characteristics shown in Table 1. Since sensor nodes are full-fledged computing devices with multiple sensing functions (such as temperature, light, and fluid flows), sensor networks are distributed systems with thousands of nodes. However the inherent properties of sensor networks differentiate them from traditional distributed systems. These distinguishing properties are summarized as follows.

Table 1: Hardware Characteristics of MICA Mote

Processor	4Mhz, 8bit MCU (ATMEL)
Storage	512KB
Radio	916Mhz Radio
Communication Range	100 ft

- Limited power: Currently, the energy of most sensor nodes is supplied by ordinary batteries. Hence, energy usage is an important factor for application design.
- High communication cost: Power consumption is dominated by radio communication. Hence, communication utilization must be minimized to conserve power.
- Low computation capability: Even though each sensor node has a micro-processor, the processing speed is much slower than modern computers. For example, MICA motes use an ATMEL processor with clock rate 4Mhz. Hence the sensor node cannot process very complex tasks. Furthermore, the memory available for programming and data is too small to store the code for complex programs and large datasets.
- Uncertainty: Readings of the sensor nodes may contain errors, which could result from environmental noises, or the inherent precision of the sensors themselves. Thus average values should be considered instead of individual readings.
- Low bandwidth: In sensor networks, sensor nodes use radio (wireless) to communicate with each other. Since the bandwidth of wireless is very low, it is necessary to decrease the number and the size of messages.

Due to these properties, existing techniques developed for traditional distributed systems can not be applied to sensor networks directly. Therefore, several techniques focused on energy and bandwidth constraints [8, 17, 18, 7, 9] have been proposed for data management and query processing over sensor networks. Since sensor nodes are powered using ordinary batteries which are expensive or impossible to replace, power-aware techniques need to be

designed to extend the lifetime of sensor networks. In this paper, we target the query processing layer, where we introduce *power-aware queries* and propose *power-aware query processing* techniques. In particular, we introduce a notion of precision into queries that allows users to tradeoff data precision and power usage. We use this notion of precision in conjunction with prediction to reduce internode communication. Moreover, we propose a technique for power-aware multi-query processing. To the best of our knowledge, this is the first work targeting multi-query processing over sensor networks. The contributions of this paper can be summarized as follows:

- We introduce the notion of precision into SQL queries.
- We propose power-aware queries over sensor networks, which gives energy-precision tradeoff to users.
- We develop a power-aware query processing technique based on *in-network aggregation* and *in-network prediction*, where aggregation and prediction take place in the network, instead of only at the basestation.
- We propose an efficient technique for power-aware multi-query processing.

The rest of the paper is organized as follows. Section 2 gives some background and motivation for power-aware query processing. The sensor network model is reviewed in Section 3. In Section 4, we introduce the power-aware query type. The power-aware query processing algorithms are presented in Section 5. In Section 6, we present multi-query processing for power-aware queries. Experimental results are shown in Section 7. We end with a conclusion in Section 8.

2 Background and Motivation

In sensor networks, queries are initiated at a basestation, where the optimized query plan is disseminated to the entire sensor network. In [7], researchers proposed the Fjords architecture for managing multiple queries over many sensors. In this architecture, each senor node sends its data directly to the basestation. This direct communication requires a large amount of transmission power which drains the node's batteries rapidly, and thus substantially decreases the lifetime of the system [16]. Minimum energy multi-hop routing protocols (MTE) were therefore proposed, in which sensor nodes route their data to the basestation through intermediate nodes [2, 14, 15, 13]. However, in general, sensor networks contain too much data for a single end user to process. Therefore data fusion, where automated methods of combining or aggregating the data into small sets of information, has been proposed [3, 5]. Data aggregation is also used to obtain more reliable data measurements by combining several unreliable data measurements to produce a more accurate signal by enhancing the common signal and reducing noise. Chandrakasan et al. [16] proposed Low-Energy Adaptive Clustering Hierarchy (LEACH), in which nodes organize themselves into clusters, one node in a cluster acts as a cluster head, and sends all the data of the nodes in the cluster to the basestation after local-aggregation. Due to local aggregation at cluster heads, the total size of the message sent by sensor nodes is reduced. In order to further reduce the power consumption of sensor networks, TAG [8] and COUGAR [17, 18] examined the properties of aggregate queries and proposed to use a tree structure, which allows data aggregation at every level of the tree for algebraic (e.g. Average) and distributive (e.g. Max/Min, Count, and Sum) aggregates. This method is referred to as *in-network aggregation*. Since a tree structure allows local-aggregation at every level of the tree, the total number of messages and the total message size in the tree structure is much less than that in the cluster based structure, where local-aggregation takes place only at cluster heads. In general, in-network aggregation can reduce the power usage by pushing part of the computation into the sensor networks.

Since the lifetime of the system is one of the most important metrics of sensor networks, the power consumption of the sensor nodes should be reduced to extend the lifetime of the entire network. Power consumption of sensor nodes is dominated by radio communication. In general, the transmission cost of sending 1 bit of data costs as much as executing 1000 CPU instructions [9]. Therefore communication needs to be reduced to save energy and prolong the lifetime of sensor networks. Several methods have been proposed to reduce the amount of communication between a data source and the base station or a server [**?**, 12, **?**, 6, **?**]. Olston and Widom [12] proposed to balance the tradeoff between precision and performance in wired-network replication management to reduce the amount of communication. In their study, the server cache stores an interval for each data source within which the exact value must be bounded. When the exact value of the data source is out of the range at the cache server, the data source sends a new interval in which the exact value must lie. A query is answered by using these intervals if they satisfy the specified precision bound, otherwise exact values need to be retrieved from the source. Recently, this work has been extended by Jain et al. [**?**], where Kalman Filter is used to predict the values of data sources. In sensor networks, Lazaridis et al. [6] proposed to compress the raw data at each sensor node, then the compressed data is sent to the basestation when the precision is out of bound. Since the

compressed data series contain less data, communication cost is reduced. They also proposed to predict the sensor readings due to the fact that the basestation has no control over when the sensor nodes deliver their compressed data. Similarly [**?**] proposed another prediction technique to monitor environment by applying MPEG techniques in prediction. In these approaches, it is assumed that there is a virtually direct communication between every sensor node and the base station, or between the data sources and the cache server. In other words, prediction only takes place in the basestation or the cache server. In sensor networks, recent works [17, 18] show significant energy savings by pushing computation into the network, which is referred to as *in-network aggregation*. With this motivation, we push prediction into network, referred to as *in-network prediction*. The idea of in-network prediction is similar to in-network aggregation, which allows prediction of partial aggregation values in sensor nodes while in-network aggregation allows to aggregate partial aggregation values. However both in-network aggregation and in-network prediction are based on tree structures, the existing two methods [12, 6] can not be directly applied to a tree structure [**?**]. Therefore we propose new techniques to realize both in-network aggregation and in-network prediction. We also propose multi-query processing technique for power-aware queries. For a single power-aware query, the energy is reduced by decreasing the number of messages sent by each sensor node. In the case of multi-queries, sensor readings and communication among different queries can be shared for further energy savings.

3 The Architecture of Sensor Networks

In this paper, we consider sensor nodes that use wireless RF radio to communicate with a basestation, which is wired to the outside world [7]. Due to the fact that low-power wireless radio has low communication distance, sensor nodes communicate with the basestation via intermediate sensor nodes. Hence there are three important components in a sensor network: a basestation, sensor nodes, and communication topology.

The basestation is a different type of node from the sensor nodes. It is not subject to the limitations of sensor nodes: power, communication, memory, and computation. It is responsible for connecting the sensor network to the outside world through wired communication. In general, users pose queries over sensor networks through the basestation. The queries are optimized and disseminated from the basestation, also the query results are assembled at the basestation and delivered to the users via the basestation.

Sensor nodes are the basic components in sensor networks. Each sensor integrated in a sensor node is a separate data source and monitors the physical environment by sampling physical signals. From another point of view, each sensor generates a discrete time series. Since uncertainty is an inherent property of sensors, the time series generated by sensors contain errors (the difference between the real environmental values and the sampled values). However more accurate results can be obtained by aggregating data from multiple sensors, i.e., summaries and aggregates of raw sensor readings are more meaningful and trustworthy than individual ones [16]. Note that the data contained at the basestation is always "stale", because the data always reflects an "old" reading of the sensor which is either delivered directly or preprocessed before it arrives at the basestation via multiple hops.

In our model, sensor nodes send and receive data using a low power wireless radio. Due to the constraint on communication distance of wireless radio (from several feet to 100 feet), data at some sensor nodes are transmitted to the basestation using a multi-hop routing protocol executed over intermediate nodes. Currently, there are several structures proposed for data collection. One simple approach is broadcasting. Each sensor node broadcasts its sampled data or relayed data from other nodes to its neighbors. Since sensor networks are completely connected, the data can reach the basestation eventually. This idea is simple and easy to implement, but is not applicable to sensor networks, because a large amount of energy and communication is consumed to send the redundant information. In this paper, we use a tree communication topology rooted at the basestation, which is adopted in [18, 9] and is the most energy-efficient communication topology. In this communication topology, each sensor node sends its data to the basestation through its ancestors in the tree, therefore synchronization of the sensor nodes along a path to the basestation is crucial for aggregation over sensor networks. We use the same synchronization method as proposed in [8, 18]. Each node synchronizes its clock using GPS. For a given sampling rate t, each node divides t into three time segments: sleeping, listening, and sending. The listening segment should be long enough to receive data from any of its children. Assume the depth of the query tree is H, t is divided into H segments (the length of each segment is t/H), and the nodes at level h will set the hth segment as their sending segment. During the sending segment, a node will send its reading or the partial aggregation results rooted at this node to its parent.

4 Power-aware Queries and Prediction

Declarative queries are the preferred way of interacting with sensor networks to aggregate data, rather than adopting application-specific procedures. This is impor-

tant since user interests may change over time and they also may need more precise answers over time. Recent works [18, 9] proposed query engines for declarative queries over sensor networks. In [18], in-network processing is proposed to decrease communication cost. Based on the inherent uncertainty of sensor nodes, we introduce precision into declarative queries, where users can specify the error tolerance range of the estimated results from the real values (which is referred to as *precision P*). We also introduce prediction that enables the basestation and sensor nodes to predict query results or partial aggregation values based on past information. In particular, the basestation or intermediate sensor nodes will only be informed about the sampled values of some sensor nodes if their predicted values may result in the query results being out of the precision bound. Hence message communication overhead can be dramatically reduced. In this section, we describe power-aware queries, then we will address some issues about prediction.

4.1 Power-Aware Queries

Current query framework proposed for sensor networks [18, 9] is based on SQL, and incorporates sampling intervals, monitoring periods into SELECT-FROM-WHERE clause. Due to the inaccuracy of single sensor readings, aggregation of multiple sensor readings is more meaningful to users. Hence we primarily focus on AVG/SUM queries. The queries have the following format, where we have introduced the *PRECISION* clause.

$$\begin{array}{l}\text{SELECT AggregationFunction}\\ \text{FROM Sensordata } s\\ \text{WHERE } s.\text{loc in } R\\ \text{DURATION } D\\ \text{EVERY } t\\ \text{PRECISION } P\end{array} \quad (1)$$

In the above query schema, $AggregationFunction$ can be aggregates AVG and SUM, s specifies the sensor types, R is the query region, D gives the query runtime, t specifies the sampling rate, and P denotes the maximum tolerated difference between the predicted values and the real values. For example, if $AggregationFunction = AVG(s.temperature)$, the semantic of this query is: Calculate the average temperature in room R with precision P, and run this query for D duration once every t time interval. Note that we use an absolute value for precision P, but our query processing technique can be easily modified to handle percentages as well.

4.2 Prediction of Time Series

For a given query, each sensor's reading is a discrete time series, where the time corresponds to the instant when the sensor value was sampled, the sensor readings are the values, and the time range is determined by the query duration D. For example, a sensor's readings at every 10 seconds during one minute are $[32.7, 33.2, 32.8, 33.5, 34.1, 33.7]$. In order to predict future readings, we need to find the trend of the time series. Due to the inherent uncertainty of sensors and due to the presence of environmental noise, it may not be possible to find the trend using a fixed-parameters model for a long time range even though each sensor's successive readings are correlated. On the other hand, from an accuracy point of view, we would like to detect the exact changing patterns of sensor readings. This, however, needs a lot of samples, high computation cost, and perhaps off-line analysis. Therefore we propose a compromise. In most cases, sensor readings will not change dramatically during a short time period, we can fit values in a short time range in a prediction model. In this paper, we propose to use the *piecewise prediction* method, where the parameters of the prediction function are adjusted to reflect the new trend of the time series if the old parameters of the function drives the predicted values out of user's tolerance range.

In our technique, the prediction function of a single sensor node is stored at the node and incoporated into the prediction function of its parent. Therefore, some sampled readings of the node do not need to be processed and transmitted to its parent when its future values can be approximately estimated from the prediction function. As a result, energy can be saved while results are guaranteed to be in a given precision range. The detailed techniques are discussed in the following section.

A method used for prediction of a time series is the state-space model [**?**]. In this model, the next state $\vec{x_{i+1}}$ is modeled as a linear combination of both the previous state $\vec{x_i}$ and some process noise u_i, which is described using the following equations:

$$\vec{x_{i+1}} = A\vec{x_i} + \vec{u_i}$$

Where A is the transmission matrix. The process measures or observations $\vec{y_i}$ are derived from the internal state $\vec{x_i}$:

$$\vec{y_i} = H_i\vec{x_i}.$$

Where H_i is the matrix relating the system state and the measurement vector. These two equations are often referred to as the process model (for prediction) and measurement model (for correction) respectively, which are the basis for all linear estimation methods, such as Kalman Filter. In sensor networks, each sensor node sends a new matrix A to its parent when the predicted value is out of the precision bound. However the *observer design problem* arises when attempting to determine the internal states of the system solely based on the knowledge of the system's outputs. In many environments there are many sources of noise for the sensor measurements which would violate the

assumption of the state-space model that the noise is zero-mean white random noise. This will degrade the quality of sensor readings (signal) which is the only source of information in the process model. As a result, the state-space model can only be used to make predictions over relatively short intervals.

One basic technique commonly used to describe a time series is linear regression [**?**], where time t is the independent variable, sensor value $v(t)$ is a variable dependent on t. Their relationship can be estimated using:

$$\hat{v}(t) = \hat{a} * t + \hat{b}.$$

Where $\hat{v}(t)$ is the estimated value of $v(t)$, a, b are two parameters referred to as the *slope* and the *intercept* respectively. One popular estimation technique of a and b is to use least square error (LST) linear fit. In the LST method, $\hat{a}$ and $\hat{b}$ are computed to minimize the residual sum of squares $\sum t = t_s t_e v(t) - \hat{v}(t)^2$, t_s and t_e are the starting time and ending time. Assuming W previous readings: $[v_1, v_2, ..., v_w]$, $\hat{a}$ and $\hat{b}$ can be computed as follows [**?**].

$$\hat{a} = \frac{\sum(t_i - \overline{t})(v_i - \overline{v})}{\sum(t_i - \overline{t})^2}.$$

$$\hat{b} = \frac{1}{W}(\sum(v_i) - b_1 \sum(t_i)).$$

Where $\overline{t}$ denotes the average of t, $\overline{v}$ denotes the average of $v_1, v_2, ..., v_w$.

Other prediction techniques have been proposed in the literature, and our proposed power aware query processing algorithms given in next section are not affected by the different prediction techniques. In this paper, compared to the state-space model, we choose linear regression for the following reasons:

- It is simple and easy to compute, which is suitable for sensor nodes with limited computation capability. In the state-space model, the computation is more complex than that of the linear regression.
- The model contains less parameters, hence it will consume less communication bandwidth and memory. In the state-space model, state transmission matrix A needs to be sent to upper level of a query tree.
- It can simulate the changing signal using piecewise linear regression (or piece-wise prediction) to guarantee a given precision. In the state-space model, since the actual state transform model is completely unknown, it is impossible to use the model for long-term prediction.

5 Power-Aware Query Processing

Users initiate queries at the basestation where a query is optimized and the query execution plan is disseminated over the sensor network. After dissemination of the query to the sensor nodes whose readings are needed to answer the query, *a query tree* is constructed over the sensor nodes. Once the query is disseminated and the query tree is built, each sensor node sends the sampled readings or partial aggregate values to the basestation through this query tree. During data collection, an interior node in the query tree should send its partial results after hearing from all of its children in order to send as few messages as possible. To achieve this, we use the synchronization method introduced in Section 3.

To execute power-aware queries, a naive and straightforward approach is as follows. Each sensor node collects its sampled data, constructs its prediction function based on these sampled data, and sends the most recent reading and the corresponding prediction function to the basestation using the query tree. When its predicted value at the basestation differs from the sampled value of this sensor node by more than the precision P, the sensor node recalculates its prediction function based on its new sampled data and updates the basestation with the new information.

Alternatively, the answer of aggregate queries can be calculated from partial results calculated in-network, thus decreasing the communication cost. However, the naive approach does not consider in-network aggregation. Thus, the average number and size of messages per sensor node in this approach is large, which will result in more energy usage and the lifetime of the sensor network is shortened. In order to improve the energy usage of power-aware queries, we propose a power-aware query processing technique which considers in-network aggregation and in-network prediction, where every sensor node n in the query tree predicts the partial aggregation values of the subtree rooted at node n.

Since we introduce precision into queries, the core of our query processing is required to preserve the precision of different kinds of aggregate queries in order to answer them using in-network aggregation and in-network prediction. We will first discuss the base case: a query tree with two levels. Then we will generalize the base case to a query tree with multiple levels.

5.1 Base Case

As mentioned in Section 2, the techniques proposed in [12, 6] can be applied to a two-level structure: from sensor nodes to the basestation. In this section, we review these two techniques and argue that they are not applicable to more than two levels. Then we introduce our technique.

Olston and Widom [12] addressed the problem of balancing the tradeoff between precision and performance for querying replicated data in replication management. In order to improve performance and reduce communication, instead of storing the exact value of an item, the server stores an interval for each item's value within which the current value must be located, as shown in Figure 1(a). N_0 denotes the server, N_1 to N_3 are sources, where item_1 to item_3 are located. Figure 1(a) shows an example where the exact values of item_1 to item_3 are 5, 11, and 10 respectively. At the server side, the intervals [3, 6], [6, 10], and [9, 12] are stored for item_1 to item_3. Queries are answered immediately using these intervals. On the other hand, the data source will update its interval at the server if its current value is not located in the old interval. For example, if item_2 has a new value 11, then it will send a new interval which contains 11 to server N_0.

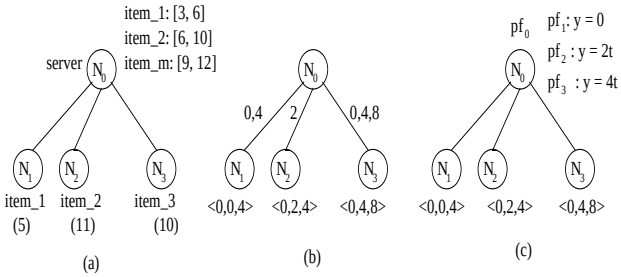


Figure 1: Comparison of different schemes for saving communication cost

Lazaridis et al. [6] proposed to use a compression technique, referred to as Poor Man's Compression - Middle Range (PMC-MR), to reduce communication cost in sensor networks. Each sensor node compresses sampled values, which can be viewed as a time series, as follows: The sensor node divides this time series into segments where the median of the sampled values in a segment is more than precision P away from the most recent sampled value. After constructing each segment, the median of the sampled values in that segment is sent to the basestation as a representative of the values in that segment. Figure 1(b) shows an example, where P is set to 3. The series of samples at node N_1 is divided into two segments, then values 0 and 4 are sent to parent node N_0. A similar analysis can be applied to the series at N_2 and N_3. The authors also proposed to use prediction techniques at parent node N_0 to predict its children's values, since the sending time of children is not known.

Even though these two proposed methods can reduce communication cost, they are not applicable for a communication topology with more than two levels, where in-network aggregation plays an important role in saving energy. In these methods, during in-network aggregation the errors of partial aggregation values will be accumulated and propagated to the basestation which may result in the query results being out of precision bound.

Before presenting our general query processing algorithm, we use the example shown in Figure 1 to introduce our method for the base case. Using the linear regression model presented in Section 4.2, each sensor node has its own prediction function. Initially, child sensor nodes send their prediction functions to their parent sensor node. For example, as shown in Figure 1(c), for the sake of simplicity, we use two readings to construct prediction functions. Prediction functions pf_1, pf_2, and pf_3 of nodes N_1, N_2, and N_3 are constructed and sent to node N_0. Assume that the precision given by users is P, node N_0 uses pf_1, pf_2, and pf_3 to predict the values at nodes N_1, N_2, N_3. At the same time, node N_1, N_2, N_3 also predict their values and sample the exact values with their sensors. If the exact value of any node differs from its predicted value by P, the node will recompute its prediction function and send both the new function and the exact value to node N_0. Otherwise, no communication is needed. For example, the predicted value of N_1 at time 3 is 0 using its prediction function $y = 0$, however, the current reading is 4 (assume $P = 3$). Thus node N_1 will recalculate its prediction function and send a new prediction function and the new value, 4, to its parent N_0.

5.2 Query Processing Algorithms

In this section, we first introduce the generic power-aware query processing algorithm. Then we will show how this generic algorithm can be used for AVG/SUM queries.

Generic Algorithm

Users query a sensor network by posing a query at the basestation. Assume the basestation has distributed the query to the sensor nodes involved in the query and the query tree has been built. Each sensor node will execute the generic algorithm, given in Algorithm 1, with the parameters specified in the query, i.e., query type, precision P, duration D, and sampling rate t.

In Algorithm 1, each node n receives partial aggregate values of its children and their corresponding prediction functions. Based on these values, it calculates the partial aggregate value and a prediction function for the subtree rooted at this node, T_n. Then for each t during D, n calculates the new partial aggregate value for subtree T_n and sends this value and the new prediction function to its parent P_n when the partial aggregate value predicted for T_n at P_n is out of precision bound. Note that the sensor node does not have to listen to its children. Instead if there is any message from its children before timeout, it listens, otherwise it sleeps after timeout. This is because energy usage is higher in listening than in sleeping. Node n also needs to compute a prediction function pf_n. This is dependent

Algorithm 1 Generic Query Processing Algorithm

Input:
T: the query tree;
n: the node executing the algorithm;
H: the height of the query tree T;
h: the height of the subtree rooted at node n;
Procedure:
//T_n is the subtree rooted at node n;
//i is a child of node n in T, T_i is a subtree rooted at i;
//v_i is partial aggregate value for subtree T_i;
//pf_i is the prediction function for v_i;
for Every t during D **do**
  Sleep(close RF) until listening time of node n;
  Set the timeout and start listening;
  if Any message before timeout **then**
    Receive the message;
  end if
  Calculate v_n and new pf_n based on the received messages;
  Calculate v_{pf_n} based on the old pf_n which was sent to n's parent;
  if $\mid v_n - v_{pf_n} \mid > P$ **then**
    Wait until its sending time;
    Send v_n and the new pf_n to the parent;
    End Procedure
  end if
end for

on the aggregation function.

In the following, we show how AVG/SUM can be executed using Algorithm 1. We also show, for a given precision P, how the precision is preserved through the partial aggregate results.

AVG and SUM queries

Algorithm 2 Calculate Subroutine for AVG

Input:
$1, 2, \ldots, m$: the children of node n;
pf_i: the prediction function for average value of subtree rooted at node i;
pf_n is the prediction function for average value of the subtree rooted at node n;
$\overline{pf_n}$ is the prediction function for average of node n;
n_i is the number of children in the subtree rooted at node i;
v is the current sampled value at node n;
Procedure:
//v_i will take the value sent by child i if it sent, otherwise it is computed from pf_i;

$$pf_n = \frac{\sum_{i=1}^{m} pf_i \times n_i + \overline{pf_n}}{m+1};$$

$$v_n = \frac{\sum_{i=1}^{m} v_i \times n_i + v}{\sum_{i=1}^{m} n_i + 1};$$

End Procedure
Output:
return pf_n and v_n;

The Calculate() subroutine for the AVG is given in Algorithm 2. The prediction function for the average value of the subtree rooted at node n is computed by taking the average of the prediction functions of all its children and itself. P in Algorithm 1 is the same P specified in the AVG query. The following theorem demonstrates that the final results of the AVG query are within the precision P.

Theorem 1 *Given an AVG query with precision P, Algorithms 1 and 2 ensure that the query answer is within P.*

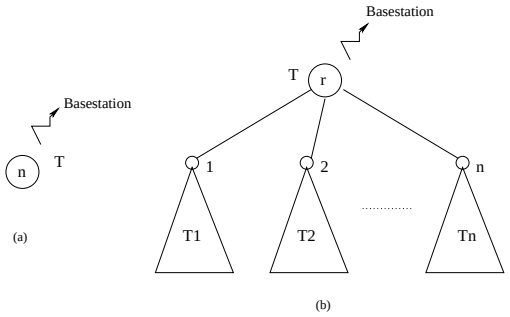


Figure 2: Proof of Theorem 1

PROOF: We use induction proof method. Base case: the level of the query tree T is one (only node n in T). Because the prediction function of node n is the prediction function of tree T, if the prediction value of n is out of the precision, the real sampling value will be used. Thus, the statement is true.

Induction Hypothesis: Assume that the statement is true when the level of the spanning tree is no larger than k, i.e., the results at the root differ from the exact results at most by P.

Induction step: As shown in Figure 2, let T be a query tree with $k+1$ levels, r be a root in T, $1, 2, ..., n$ be direct children of r, $T_1, T_2, ..., T_n$ be subtrees rooted at nodes $1, 2, ..., n$ respectively, $v_1, ..., v_n$ be the average values for subtrees $T_1, T_2, ..., T_n$, v_r be the current reading of root r and $pf_1, pf_2, ..., pf_n$ be the prediction functions for these values. Based on Algorithm 2, the prediction function for the average value of tree T is:

$$pf_T = \frac{\sum_{i=1}^{n}(pf_i \times n_i) + pf_r}{\sum_{i=1}^{n}(n_i) + 1}.$$

Where pf_T denotes the prediction function for average value of tree T, and n_i is the number of children of node i. Thus, the predicted average value of tree T is:

$$v_{pf_T} = \frac{\sum_{i=1}^{n}(v_{pf_i} \times n_i) + v_r}{\sum_{i=1}^{n}(n_i) + 1}.$$

Where v_{pf_i} is the average value based on the prediction function of subtree T_i. The exact average value of tree T is:

$$v_T = \frac{\sum_{i=1}^{n}(v_i \times n_i) + v_r}{\sum_{i=1}^{n}(n_i) + 1}.$$

Based on the induction hypothesis, since each subtree T_i has at most k levels, it satisfies:

$$\mid v_i - v_{pf_i} \mid \leq P.$$

Therefore,

$$\mid v_T - v_{pf_T} \mid = \mid \frac{\sum_{i=1}^{n}((v_i - v_{pf_i}) \times n_i) + (v_r - v_r)}{\sum_{i=1}^{n}(n_i) + 1} \mid$$

$$\leq \frac{P\sum_{i=1}^{n}(n_i)}{\sum_{i=1}^{n}(n_i)+1} \leq P.$$

Thus

$$\mid v_T - v_{pf_T} \mid \leq P.$$

Therefore, the answer of the AVG query, which is executed using Algorithms 1 and 2, satisfies the precision P.□

The query processing of SUM queries is similar to AVG queries. However, we have to modify the generic algorithm and calculate subroutine for SUM. Line 19 of Algorithm 1 should be $n_n/N \times P$, where N is the total number of nodes in the query tree and n_n is the number of nodes in the subtree rooted at node n, P is the precision specified in SUM queries. The calculate subroutine returns the sum of the values of all nodes in subtree rooted at node n and the prediction function associated with that sum. The correctness proof of SUM query processing is similar to that of AVG query processing. Due to the space limitation, we do not include the details in this paper.

6 Multi-Query Processing

Users typically impose multiple queries over a sensor network at the basestation during a time range (a batch of queries). This could happen, for example, when a set of sensors detect some anomalies and need to learn the average temperature of some specified regions. Current proposals [8, 18, 9] for query processing over sensor networks do not consider multi-query processing. The problem of multi-query optimization has been studied in different contexts, such as relational database systems(RDBMs) and data streams. In RDBMs, multi-query optimization aims to share common sub-expressions [**?**, **?**] among different queries. In data streams [**?**], the goal of multi-query optimization is to share processing among different queries with the assumption that the data is already collected. However multi-query processing in sensor networks is motivated by sharing sensor readings and data transmission among different queries if the query regions of different queries intersect. Therefore, energy can be conserved by sharing the results of common regions among different queries.

For the sake of simplicity, we assume multiple queries run as a batch and all queries have the same precision and the same sampling rate. If two intersecting queries have different sampling rates, the sensor nodes in the common regions choose the higher sampling rate since the sensor readings with higher sampling rates can give better approximation of the real environment. Similarly, if two intersecting queries have different precision, the sensor nodes in the common regions choose the smaller value, which corresponds to the higher precision. This guarantees that the results of each query satisfy the specified precision in this query. Let $Q_1, Q_2, ..., Q_n$ be queries over regions $R_1, R_2, ..., R_n$. Note that we can assume that all queries have the same type of aggregation functions, because the SUM can always be approximated by AVG*N where N is the number of sensor nodes involved. A query tree T_{union} is first constructed over the union of queried regions $R_1 \cup R_2 ... \cup R_n$. Then queries are answered at the basestation using partial results from some of the involved nodes in the union region. For example, Figure 3 shows a query tree built over $R_1 \cup R_2$ and is used to answer both Q_1 and Q_2.

To answer a query in the batch, we can not use the single query processing algorithms presented in Section 5, since the query tree of multiple queries is built over the union of all query regions instead of a query tree for each query. Each query region R_i of a single query Q_i intersects with the query tree T_{union}. The query region is treated as a black box, which crosses some edges of the query tree T_{union}. Assuming in-network aggregation, the query answer can be answered using information contained in the messages sent on the edges intersecting with R_i. For example, the answer for query Q_1 can be computed by subtracting partial aggregation values at node 3 and 4 from the partial aggregation values at node 1. Given $Q_1, Q_2, ..., Q_n$, for a query Q_i, we classify the edges which intersect the query region Q_i in the query tree T_{union} into the following types.

Definition 1 *Incoming edge: A directed edge $< e, f >$ is an incoming edge if it intersects with the query region R_i, and the ending node f is inside the query region and the starting node e is outside the query region.*

Definition 2 *Outgoing edges: A directed edge $< e, f >$ is an outgoing edge if it intersects with the query region R_i, and the starting node e is inside the query region and the ending node f is either outside the query region or is the basestation.*

For example, in Figure 3, e_1 is an outgoing edge and node 1 is the starting node of e_1. e_2 and e_3 are the incoming edges where node 6 is the ending node, nodes 3 and 4 are the two starting nodes. Note that there can be many incoming edges and outgoing edges for a query. The incoming and outgoing edges are determined when queries are disseminated over the sensor networks. Each node knows whether it is the starting node of an incoming edge or an outgoing edge when the query tree is constructed. This only needs a small modification to the query tree construction algorithm of a single query. After the basestation disseminates all queries, every sensor node n sends its query list l_n to its parent p, which consists of all queries involving n. Then, the parent node p compares the query list of n with its own query list l_p. If a query Q_i is in l_p but not in l_n, it means that the edge $< n, p >$ is an incoming edge of query Q_i. Then node p sends to node n a message which

states that node n is the starting node of an incoming edge and its partial aggregation values need to be propagated to the basestation. Similarly, if a query is in l_n but not in l_p, it means that the edge $< n, p >$ is an outgoing edge and node n will be notified too.

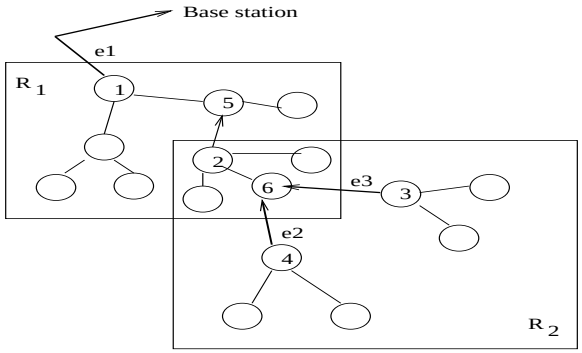


Figure 3: Power-aware multi-query processing algorithm

Since all queries have the same sampling rate, the sensor nodes only need to sample once each time interval to answer all queries and execute a query processing algorithm given in Algorithm 1 except for a minor change to the messages passed among sensor nodes. In order to answer query Q_i, the partial aggregation values and prediction functions at the sensor nodes which are the starting nodes of the incoming or outgoing edges of region R_i need to be sent to the basestation. Therefore the starting nodes of the incoming and outgoing edges of R_i need to inform their parents to pass their partial aggregation values and prediction functions to the basestation. Consider a node p which is the ending node of an incoming edge $< e, p >$, node p, in addition to computing the partial aggregation values and prediction functions of the subtree T_p rooted at the node p, also transfers the partial aggregation values of node e along with its own partial aggregation values and prediction functions. We use the same example in Figure 3. Nodes 3 and 4 inform node 6 to pass their partial aggregation values and prediction functions, PA_3 and PA_4, to the basestation, and node 6 first uses PA_3 and PA_4 to compute its own partial aggregation values and prediction functions PA_6 of the subtree rooted at node 6, and then transfers PA_3, PA_4, and PA_6 to node 2 which is the parent of node 6. For the basestation to answer queries, each partial aggregation value should contain information that it is a value of a starting node of an incoming edge or an outgoing edge of some query. This can be done by adding the starting and ending nodes' numbers of an incoming or an outgoing edge to the message which contains the partial aggregation values and prediction functions at the starting node. For the example shown in Figure 3, the message sent by node 4 to node 6 contains the partial aggregation values and prediction functions at node 4 and node numbers 4 and 6. When the basestation reads this information, using the the knowledge of positions of all sensor nodes, it determines that edge $< 4, 6 >$ is an incoming edge for query Q_1. Based on the synchronization at the end of each time interval, the basestation executes Algorithm 3 to answer each query Q_i in the batch using the partial aggregation values and prediction functions from the outgoing and incoming edges. The results for Q_i satisfy the precision constraint specified in Q_i, which is proved by the following theorem.

Algorithm 3 Calculate the results for query Q at the basestation

Input:
$1, 2, \ldots, n_{out}$: the outgoing edges;
$1, 2, \ldots, n_{in}$: the incoming edges;
pf_i: the prediction function for average value of subtree leaded by outgoing edge i;
pf_j: the prediction function for average value of subtree leaded by incoming edge j;
pf_Q is the prediction function for average values of query Q;
n_i is the number of children in the subtree leaded by outgoing or incoming edge i;
Procedure:
//v_i will take the value sent by subtree i if it sent, otherwise it is computed from pf_i;
$pf_Q = \frac{\sum_{i=1}^{n_{out}} pf_i \times n_i - \sum_{j=1}^{n_{in}} pf_j \times n_j}{\sum_{i=1}^{n_{out}} n_i - \sum_{j=1}^{n_{in}} n_j}$;
$v_Q = \frac{\sum_{i=1}^{n_{out}} v_i \times n_i - \sum_{j=1}^{n_{in}} v_j \times n_j}{\sum_{i=1}^{n_{out}} n_i - \sum_{j=1}^{n_{in}} n_j}$;
End Procedure
Output:
return pf_Q and v_Q;

Theorem 2 *Algorithm 3 guarantees that v_Q is the result of query Q and is within the precision range.*

PROOF: We first need to prove that v_Q is the query result, i.e., v_Q only includes the sensor node values (the real readings or the predicted values) in the query region. Each subtree T_{in} rooted at the starting node of one incoming edge e_{in} must be included in a subtree T_{out} rooted at the starting node of an outgoing edge e_{out}, therefore the values of the subtree T_{out} contains the values of T_{in}. The above formula for computing v_Q is the subtraction of the sum of all outgoing edges' values and the sum of all incoming edges' values, which only contains the values inside the query region.

For a given query Q, in Algorithm 3, if there is no update from its outgoing and incoming edges, the predicted result of query Q, v_{pf_Q}, is computed using the predicted values v_{pf_i} of its outgoing and incoming edges:

$$v_{pf_Q} = \frac{\sum_{i=1}^{n_{out}} v_{pf_i} \times n_i - \sum_{j=1}^{n_{in}} v_{pf_j} \times n_j}{\sum_{i=1}^{n_{out}} n_i - \sum_{j=1}^{n_{in}} n_j};$$

Where v_{pf_i} is the predicted value of subtree i and is computed using pf_i. The exact value v_{T_Q} of Q is:

$$v_{T_Q} = \frac{\sum_{i=1}^{n_{out}} v_{T_i} \times n_i - \sum_{j=1}^{n_{in}} v_{T_j} \times n_j}{\sum_{i=1}^{n_{out}} n_i - \sum_{j=1}^{n_{in}} n_j};$$

Where v_{T_j} is the exact average value of outgoing or incoming edge subtree. Since the difference of nodes in all outgoing edge subtrees and in all incoming edge subtrees of query Q is all nodes in query Q, and each node k in query region of Q satisfies:

$$\mid v_{T_k} - v_{pf_k} \mid \leq P.$$

Where v_{T_k} is the exact value of node k, v_{pf_k} is the predicted value of node k, P is the precision given in Q. Assume the number of involved nodes in Q is n_Q ($= \sum_{i=1}^{n_{out}} n_i - \sum_{j=1}^{n_{in}} n_j$), we have,

$$\begin{aligned} &\mid v_{T_Q} - v_{pf_Q} \mid \\ &\quad = \frac{\sum_{i=1}^{n_{out}} (v_{T_i} - v_{pf_i}) \times n_i - \sum_{j=1}^{n_{in}} (v_{T_j} - v_{pf_j}) \times n_j}{\sum_{i=1}^{n_{out}} n_i - \sum_{j=1}^{n_{in}} n_j} \\ &\quad = \frac{\sum_{k=1}^{n_Q} (v_{T_k} - v_{pf_k})}{n_Q} \leq \frac{\sum_{k=1}^{n_Q} P}{n_Q} \leq P. \square \end{aligned}$$

7 Experimental Evaluation

In this section, we present experimental results for power-aware query processing. We compare our power-aware query processing technique with TAG approach [8] and the naive approach. Note that TAG only considers in-network aggregation, i.e., all queries are executed without considering precision. Therefore, experimental results for $P = 0$ is for TAG approach. In the naive approach, each sensor computes and sends its prediction function to the basestation using the query tree.

In our experiments, we use the tree based communication topology. We assume all nodes are well synchronized. In the naive approach, each node uses the synchronization to merge the packets from its children. In power-aware query processing, we use this synchronization for in-network aggregation and in-network prediction.

For all queries, we compare the average number of bits sent by a single sensor node because the communication cost is directly determined by the number of bits sent. In our experiments, we use the linear regression as a prediction model. This results in less computation cost. In general, the computation cost of the prediction function of one sensor node is the cost of sending $1/4$ bit. Due to the nature of prediction and in-network prediction, prediction functions of sensor nodes do not need to be recomputed most of the time. Hence we only consider communication cost in our experiments.

7.1 Experimental Setup

In our experimental scenario, we place N^2 sensor nodes over an $N \times N$ grid with each node located at a grid point. Each node can communicate with its eight direct neighbors on this grid (excluding the sensor nodes at the border of the grid).

We conducted experiments on power-aware query processing for AVG and SUM queries over a synthetic dataset and real temperature and humidity datasets from the Tropical Atmosphere Ocean Project [11]. The synthetic data is generated by a random walk:

$$x[0] = 20 \text{ and } x[n] = x[n-1] + s_n, \text{ where } s_n = \mp 0.25.$$

The characteristics of the datasets are shown in Figure 4. Figure 4(a) shows the synthetic dataset we generated, the real dataset for humidity and temperature are displayed in Figures 4(b) and (c). The queries we execute are as follows:

$$\begin{array}{l} \text{SELECT AVG/SUM} \\ \text{FROM Sensordata } s \\ \text{WHERE } s.\text{loc in GRID } N \times N \\ \text{DURATION } D \\ \text{EVERY } 1 \\ \text{PRECISION } P \end{array} \quad (2)$$

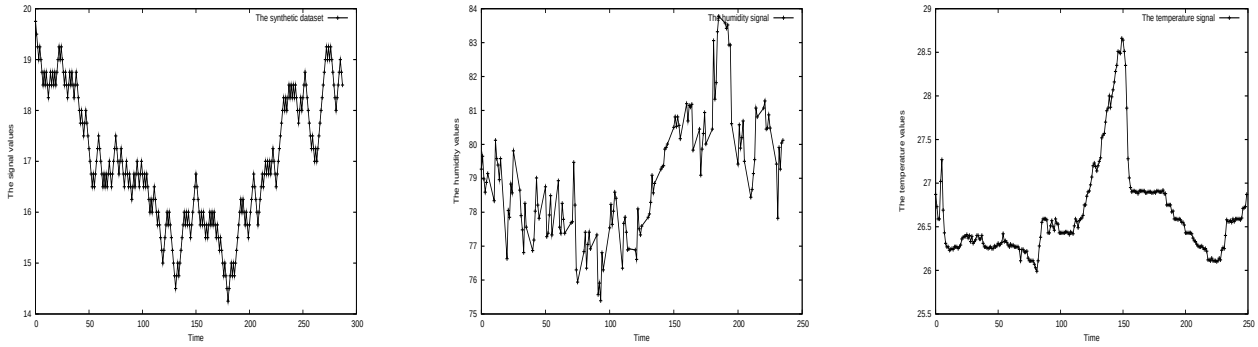
We vary D and P for different queries. For the synthetic dataset, D is always equal to 1000 while for the temperature and humidity datasets D is set to 250. The reason for choosing different values for duration D is because the real datasets only contain nearly 250 readings per sensor node. However, in order to see the long-run gains of our technique, we choose D as 1000 in the experiments over the synthetic data.

7.2 Experimental Results For Single Query Processing

In this section, we present the experimental results for single query processing over the synthetic and real datasets.

7.2.1 Synthetic Dataset

Figure 5 shows the experimental results over the synthetic dataset for AVG queries. The Y axis represents the size of the query region (or grid), which also determines the number of sensor nodes involved in the queries. For example, if the region size is 10×10, we query 100 sensor nodes to compute their average values. We set different levels of precision ($P = 0, 0.25, 0.5, 0.75, 1$). The results show that the larger the precision P, the less messages are needed to answer the query. Since $P = 0$ is the case of TAG approach, TAG always has worse performance than our power-aware query processing technique. This is because our technique considers both in-network aggregation



(a) (b) (c)

Figure 4: The data characteristics of synthetic and real datasets

and in-network prediction. From Figure 5, we observe that the average number of bits sent by a node decreases when $P > 0$ and the diameter of the network increases. This is due to the fact that using in-network prediction, the predicted partial aggregation values at the higher levels of the tree have more chance to be in the precision range since the change resulted from increased values of some nodes can be reduced by the changes resulted from decreased values of some other nodes. Furthermore, a tight precision range, e.g., 0.25, results in significant savings in energy when compared to queries without any precision, i.e., when $P = 0$.

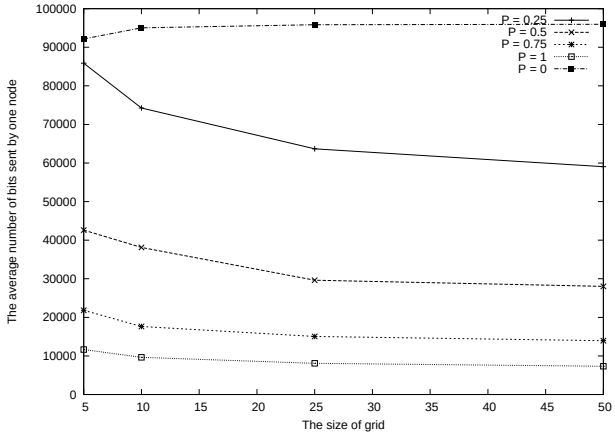


Figure 5: The average number of bits sent by each node for AVG queries

In order to show the effectiveness and scalability of in-network computing, we compare the two query processing techniques: the power-aware approach and the naive approach. The results for increasing the grid sizes are shown in Figure 6, which are based on the synthetic dataset. For a given precision P, power-aware query processing methods outperform the naive approach by more than two-fold savings in terms of the average bits sent per node. We conclude that power-aware query processing is much more scalable than the naive approach because the naive approach only considers prediction without considering in-network aggregation and in-network prediction. Based on the comparison of these two approaches over AVG queries, we observe that the power-aware query processing approach always outperforms the naive approach. In the naive approach, when the diameter of the network increases, the average number of bits sent per node increases linearly, as shown in Figure 6. This is again because there is no in-network aggregation. Similar results are observed for SUM queries.

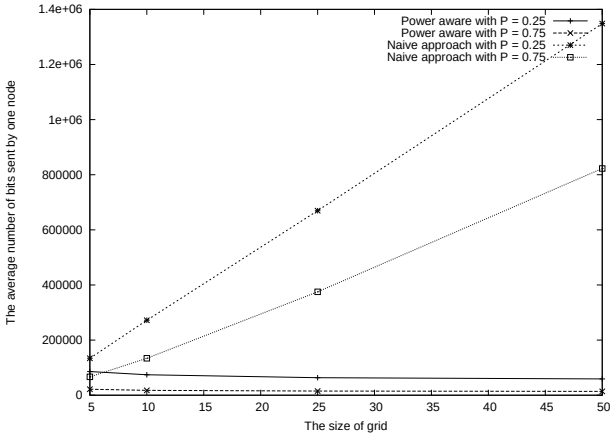


Figure 6: The comparison of power-aware query processing and the naive approach

Figure 7 shows the results for SUM queries over the synthetic dataset. With the increase in diameter of the sensor network and the number of sensor nodes, the average number of bits sent increases. Since the results of SUM queries

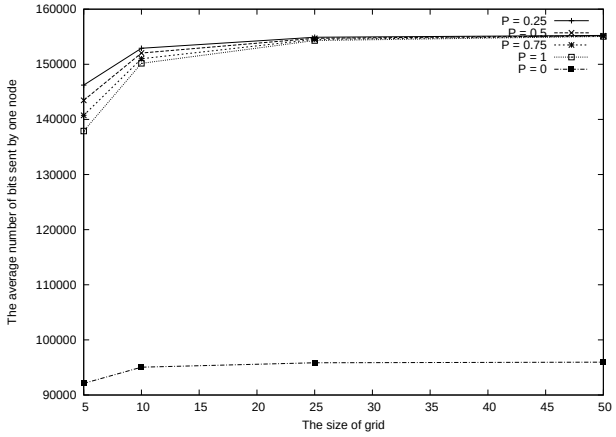


Figure 7: The average size of bits sent by each node for SUM queries

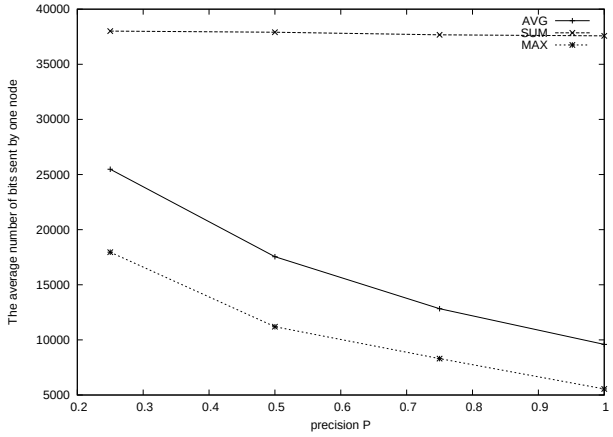


Figure 9: The experimental results over humidity data

are the addition of values of all sensor nodes, the query results are affected by the value of any sensor node. The results of the addition of all sensor values is a much larger value than the value of any sensor node. However, the results use the same precision as the one of a single sensor, hence the results with the very small precision value are almost the same as their absolute values. In another words, sensor nodes almost need to send the parameters of their prediction function with a very small value of P while sensor nodes only need to send their sampled values when $P = 0$. This explains that the average number of bits sent of each node when $P = 0$ is less than the other cases, as shown in Figure 7. In conclusion, for SUM queries, precision P should be set as a percentage of the results.

7.2.2 Real Dataset Experiment

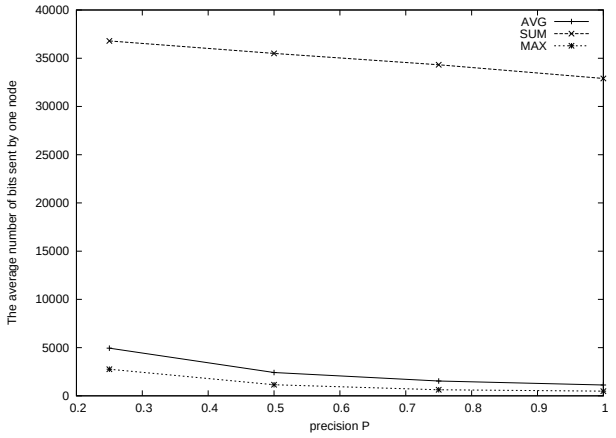
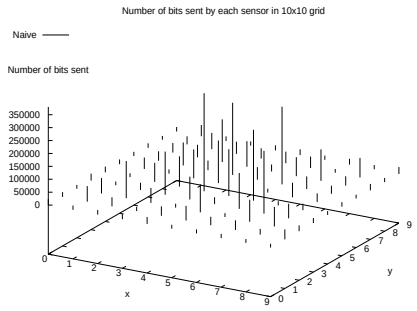


Figure 8: The experimental results over temperature data

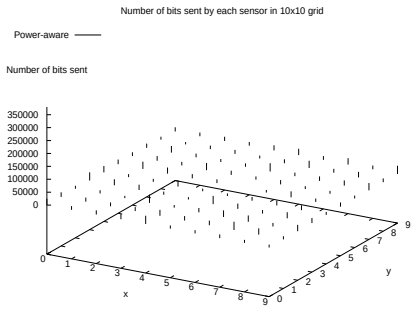
In order to show the effectiveness of our query processing techniques, we tested them over real temperature and humidity data from Tropical Atmosphere Ocean Project [11]. The characteristics of the data is shown in Figures 4(b) and (c). The experiments are conducted over 10×10 grid with 100 sensor nodes. Figure 8 shows the results of different query types over temperature data by varying the precision. The larger the precision range, the less number of bits are sent.

Similarly, Figure 9 shows the results of different query types over the humidity dataset by varying the precision. The number of messages in AVG queries decrease significantly when the precision is increased. However, the SUM query does not change significantly as discussed before. Compared to the results over the temperature dataset, the larger change in number of messages over the humidity dataset is due to the larger fluctuations in the humidity dataset, which can be observed from Figure 4. In another words, the prediction functions in processing the humidity data are updated more often than the ones over the temperature dataset.

We also conducted an experiment to test the distribution of the total bits sent over all sensor nodes. The results are shown in Figure 10. The X and Y axes denote the 10×10 grid and 100 sensor nodes. The Z axis is the total number of bits sent by each sensor node. We observe that our query processing methods can achieve balanced energy consumption compared to the naive approach. Since we placed the basestation at the center of the grid, the sensor nodes which are near to the basestation in the naive approach have more bits to send, as shown in Figure 10(a). Thus these nodes will drain their power more quickly. On the other hand, in the power-aware query processing approach, the energy usage of sensor nodes is balanced, as shown in Figure 10(b). Hence, as the network diameter is increased, our approach will be much more scalable when compared to the naive approach.



(a) Power-aware query processing



(b) Naive approach

Figure 10: The number of bits sent by each sensor node with precision 0.5 of AVG queries over a 10×10 grid

7.3 Experimental Results for Multi-Query Processing

In order to show the effectiveness of our proposed multi-query processing technique (MPO), we compare it with the following two multi-query processing techniques.

- **Naive multi-query processing approach (NMP):** The Basestation collects the readings of all sensor nodes whose readings are needed to answer queries. Then the basestation computes the results for each query. This method does not consider in-network aggregation. In terms of collecting messages from sensor nodes, we use tree structures which have been proposed to allow merging packets in [17], hence communication cost is reduced by reducing packet header overhead.

- **Power-aware query processing without optimization (MPWO):** The basestation processes each query separately using a single power-aware query processing technique, i.e., processing each query independently using its own tree and employing both in-network aggregation and in-network prediction.

In our experiments, we constructed a set of random AVG queries over a 50×50 grid and varied the total number of queries from 10 to 100 to measure the average number of bits sent by a sensor node. All queries have $P = 0.5$ and $D = 300$. The experiments are conducted over the temperature datasets. Given $N_Q + 1$ queries, the query set consists of $N_Q/3$ queries over 15×15, $N_Q/3$ queries over 20×20, $N_Q/3$ queries over 25×25, and a single query over 50×50.

The experimental results are given in Figure 11, which shows that our multi-query processing technique significantly outperforms the other two approaches. Multi-query

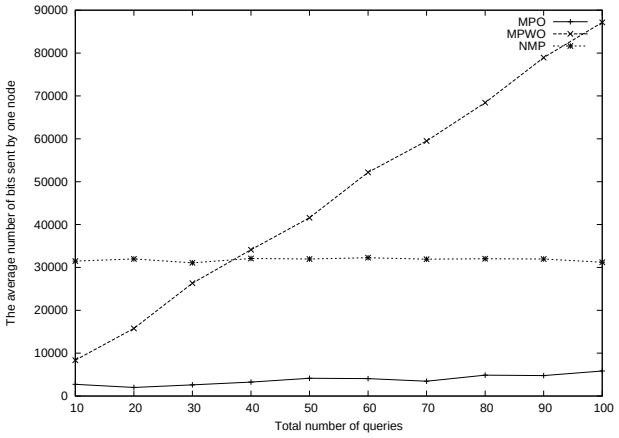


Figure 11: Power-aware multi-query processing algorithm

processing improves the communication cost by one order of a magnitude compared to the naive multi-query processing approach. The power-aware query processing without optimization becomes very inefficient and has worse performance than the naive multi-query processing technique when the number of queries increases. Since in power-aware query processing without optimization, a tree is built for each single query, one sensor node sends its readings multiple times if it is involved in multiple queries. However in the naive multi-query processing approach, a single tree is built over the entire sensor network, hence each sensor node only sends its readings once without considering the number of queries. Therefore, even though the power-aware query processing without optimization approach benefits from in-network aggregation, the cost of sending each sensor readings multiple times is larger than the savings from in-network aggregation when the number of queries increases. Hence, it is not scalable when compared to the other two methods. From Figure 11, we

observe that the number of bits sent in the multi-query processing technique slightly increases with the number of queries, because the number of incoming and outgoing edges increase with the larger number of queries.

8 Conclusion and Discussion

In this paper, we introduced a precision based framework for query processing over sensor networks. We proposed a power-aware query processing technique which incorporates in-network aggregation and in-network prediction. Our experimental results show that our query processing technique can reduce the number of message (or the size of bits) sent by each sensor node dramatically. Furthermore, the energy usage of sensor nodes is balanced, which means that our technique can be applied to large-scale sensor networks. We also proposed multi-query processing over sensor networks by sharing the readings and communication of common sensors among different queries. The energy saved from the sampling and communication prolongs the lifetime of the sensor network.

In this paper, even though we do not cover node failures, link failures and changes, our proposed techniques are not affected by these issues. Since these problems can be solved by periodical heart beat messages sent by the parent sensor nodes. If a parent node does not receive the replies from a child, it assumes that either the child node is failed or changes its parent node. Then the parent node will remove the prediction functions and partial aggregation values of this child node.

References

[1] A. Cerpa, J.Elson, D. Estrin, L. Hamilton, and J. Zhao. Habitat monitoring: application driver for wireless communications technology. *In ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, 2001.

[2] M. Ettus. System capacity, latency and power consumption in multihop-routed ss-cdma wireless networks. *Radio and Wireless Conference (RAWCON '98)*, August 1998.

[3] D. Hall. Mathematical techniques in multisensor data fusion. *Artech House*, 1992.

[4] J. Hill and D. Culler. A wireless embedded sensor architecture for system level optimization. *Technical report U.C. Berkeley*, 2001.

[5] L. Klein. Sensor and data fusion concepts and applications. *SPIE Optical Engr Press*, 1996.

[6] Iosif Lazaridis and Sharad Mehrotra. Capturing sensor-generated time seires with quality guarantees. *International Conference on Data Engineering (ICDE 2003)*, March 2003.

[7] S. Madden and M.J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. *International Conference on Data Engineering (ICDE 2002)*, March 2002.

[8] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. *5th Symposium on Operating Sytsem Design and Implementation*, December 2002.

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

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

[11] M. J. McPhaden. Tropical atmosphere ocean project. *Pacific marine environmental laboratory*. http://www.pmel.noaa.gov/tao/.

[12] C. Olston and J. Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. *In VLDB2000*, September 2000.

[13] Timothy J. Shepard. A channel access scheme for large dense packet radio networks. *SIGCOMM*, pages 219–230, 1996.

[14] Mike Woo Suresh Singh and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. *Mobile Computing and Networking*, pages 181–190, 1998.

[15] Meng T. and Volkan R. Distributed network protocols for wireless communication. *In Proc. IEEEE ISCAS*, May 1998.

[16] Anantha Chandrakasan Wendi Rabiner Heinzelman and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. *HICSS*, January 2000.

[17] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Records*, 31(3), 2002.

[18] Yong Yao and Johannes Gehrke. Query processing for sensor networks. *In CIDR 2003*, January 2003.