

Sensor Node Selection for Execution of Continuous Probabilistic Queries in Wireless Sensor Networks

Kam-Yiu Lam¹, Reynold Cheng², BiYu Liang¹ and Jo Chau¹

Department of Computer Science¹
City University of Hong Kong
Tat Chee Avenue, Kowloon Tong
HONG KONG

Department of Computer Sciences²
Purdue University
West Lafayette, IN 47907
USA

cskylam@cityu.edu.hk, ckcheng@cs.purdue.edu, byliang@cs.cityu.edu.hk

ABSTRACT

Due to the error-prone properties of sensors, it is important to use multiple low-cost sensors to improve the reliability of query results. However, using multiple sensors to generate the value for a data item can be expensive, especially in wireless environments where continuous queries are executed. Further, we need to distinguish effectively which sensors are not working properly and discard them from being used. In this paper, we propose a *probabilistic* approach to decide what sensor nodes to be used to answer a query. In particular, we propose to solve the problem with the aid of *continuous probabilistic query (CPQ)*, which is originally used to manage uncertain data and is associated with a probabilistic guarantee on the query result. Based on the historical data values from the sensor nodes, the query type, and the probabilistic requirement on the query result, we derive a simple method to select an appropriate set of sensors to provide reliable answers. We examine a wide range of common *aggregate queries*: average, sum, minimum, maximum, and range count query, but we believe our method can be extended to other query types. Our goal is to minimize sensor data aggregation workload in a network environment and at the same time meet the probabilistic requirement of the CPQ.

Categories and Subject Descriptors

C.4 [Performance of Systems]: Performance Attributes, Reliability, Availability, and Serviceability.

General Terms

Performance, Design.

Keywords

Sensor Network, Continuous Probabilistic Query.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSN '04, October 15, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-934-9/04/0010...\$5.00.

1. INTRODUCTION

Sensors are deployed in system environments to capture the status of physical entities. As an example, an air-conditioning control system monitors temperature readings obtained from sensors in different parts of the building, and adjusts temperature of each office accordingly. A traffic-monitoring system keeps track of traffic flow by deploying sensors in different districts of the city, in order to detect the presence of any critical issues like traffic congestion and accidents. Sensors are also commonly used in weather systems and habitat monitoring systems, collecting details such as temperature, pressure, rainfall, wind speed etc., where their values are being monitored continuously, and are used to provide up-to-date information about weather conditions.

A serious problem faced by sensors is that their readings may be erroneous and noisy [EN03, NN04]. If these readings are not accurate enough, the system may yield incorrect information to users and make false decisions. A good idea to improve the reliability of sensor readings is to employ more than one sensor to read values in the same region. For example, to monitor vibration level of a building, instead of using just one sensor, multiple sensors can be installed in the building, and the average value of these sensors is treated as the true vibration level of the building. This idea is supported by the fact that prices of sensors have been dropping significantly in recent years.

Unfortunately, deploying multiple “redundant” sensors for improved reliability causes higher consumption of network bandwidth. This is particularly troublesome in a wireless sensor system, such as smart sensor networks [N04], where sensors are connected to each other through a wireless, low cost and limited-bandwidth network. The high network communication cost implies that we cannot allow unlimited number of redundant sensors to supply their readings. Furthermore, getting readings from more sensors may not always improve the reading reliability if the readings from the additional sensors contain errors. Therefore, two important questions in querying the sensors are: (1) *what is the lowest number of sensors that can achieve a guaranteed level of accuracy?* (2) *Which sensors should be selected?* Since sensors are error-prone, in accessing the data values from multiple sensors, it is important to identify which of them generate abnormal readings, and avoid from using them. Defining an abnormal reading is non-trivial, however.

To see how these questions can be solved, let us first explain the concept of *probabilistic queries*. A probabilistic query was originally used to evaluate data inherent with uncertainty (e.g., measurement error), and it augments confidence to answers with

probabilistic guarantees [CKP03]. For example, temperature values of rooms can be modeled as Gaussian distributions, and a probabilistic query asking “Which room has a 60% chance of yielding a temperature over 20°C” will return names of rooms with a probability over 60% of satisfying the query. Here we call “60%” the *probabilistic requirement* of a probabilistic query, which can be viewed as the level of confidence required for query answers.

It is possible to use probabilistic queries as a tool to derive the *optimal* number of redundant sensors used. By collecting various readings from the entities (or physical signals) of the same region, we can derive the distribution of error for the true reading. A probabilistic query can then be evaluated on the reading (with uncertainty) and yield probabilistic answers. More importantly, the uncertainty information used to derive the probabilistic results can help us to find out which sensors are required for reliable readings. In practice, queries in such kind of a system are being evaluated for an extended amount of time (called *continuous queries*), which monitor sensor readings in a non-stopped manner. We study the continuous version of probabilistic queries, called *continuous probabilistic queries* (CPQ), which remain active and access data during a user-defined period. Two examples are:

- Tell me the highest temperature of the rooms from *A* to *D* from now to 10 min later.
- Tell me the traffic condition 1km around the central station from now to 30min later.

The continuous execution of queries makes the sensor selection problem even more critical to the computation and communication overhead for query processing. Each CPQ is associated with a probabilistic requirement. The reported query results need to meet the accuracy requirement. For example, for a range count query, it needs to indicate the number of entities within the range with the required probability specified in the query. For a query finding the average, it has to return results with bounded variance specified in the average query.

In this paper, we focus on selecting the right set of sensors for multiple sensor aggregation in order to obtain data values that are precise enough to meet the probabilistic requirement of the queries, instead of cleaning or calibrating individual sensor values. We concentrate on probabilistic queries with *aggregate* functions such as *mean*, *maximum* and *minimum*, and propose different forms of probabilistic requirements for each of them, which allow users to specify as a quality guarantee. We also analyze the impact of data on accuracy of query results for various types of aggregate queries. Based on the analyses, we propose algorithms to determine the accuracy requirements of individual data items being queried. Moreover, we propose an approach to determine the sample size and the set of sensors for multiple sensor aggregation to obtain the data item while meeting the accuracy requirements imposed on it.

This paper is organized as follows. In Section 2 we discuss related works. Section 3 describes the wireless sensor model, and the underlying sensor data and query models. In Section 4 we present our algorithms that solve the problems of sensor selection. Section 5 presents other interesting issues about this research field and Section 6 concludes the paper.

2. RELATED WORKS

Querying sensor data in wireless sensor networks, with data uncertainty taken into account, is a new area in mobile computing research. It has received growing interests in recent years as the

error inherent to sensor data can be a serious problem to the accuracy of queries. Indeed, as pointed out by [WYYE04], there is a *lower bound* of uncertainty in location measurements in wireless sensor networks. Hence sensor error must be treated carefully in order to produce accurate answers.

Probabilistic queries provide an effective way of evaluating data with uncertainty. They process uncertain data and augment confidence guarantees to query answers. The issues of data uncertainty and probabilistic queries are studied comprehensively in [CKP03]. The authors observed that sensor data are only sampled periodically, resulting in inconsistency between system data and the physical entities they model. This inconsistency, or uncertainty of an item, can be represented by a closed bound, together with a probability density function of the data value within the bound. Under this model, they classified probabilistic queries according to whether aggregate operators are involved, as well as the form of the answer. For each query class, they proposed evaluation algorithms and defined the quality of probabilistic answers. Unlike our paper which assumes a wireless sensor network environment, their system model is simple and assumes the host communicates directly with every sensor source. Their method of reducing uncertainty is by sampling hot items more frequently; our approach, on the other hand, selects appropriate sensors to improve reliability in sensor readings. Also, unlike our paper, they do not study *continuous queries*, and do not allow users to specify probabilistic requirements, which can be seen as a quality guarantee on query results.

In [EB03], Eiman and Badri proposed a Bayesian approach for reducing one important type of data uncertainty -- the random noise. They designed a framework that utilizes the Bayes' rule to reduce uncertainty and illustrate how it is used for online cleaning of noisy sensor data. They also proposed algorithms for answering queries over uncertain data. However, their proposed method does not take into consideration the confidence requirement of probabilistic queries. Their noise cleaning technique aims at cleaning random error of individual sensors, while our method selects the right sensors for aggregation in order to obtain more precise data values to meet the confidence requirement of probabilistic queries with minimum data collection cost.

Calibration errors have been addressed recently in [BMEP03]. The authors proposed a post deployment calibration approach utilizing temporal correlation between sensor pairs. They focus only on calibration and give no consideration to random noise. Moreover, their approach does not take user queries into consideration; it is just a method for calibrating sensor readings.

Kumar proposed a massive sensor framework where each sensor can communicate with each other [K03]. He introduced the concept of *micro-sensor net*, which is essentially a set of sensors fully connected with each other, one of which is selected as the leader or *coordinator*. When sensors in other micro-sensor nets intend to communicate with one of the sensors of a micro-sensor net, they must get certification from the coordinator of the corresponding micro-sensor net. This scheme provides authentication and Denial of Services (DoS) protection. Our paper employs a similar model, where each coordinator manages a group of sensors. However, our focus is on how to select sensors for result reliability.

The issues of executing aggregate queries efficiently in a sensor network are studied in [MFHH02]. They proposed a *data gathering tree* over the network structure, which is rooted at the

base station (also called *sink*). When a sink collects information from sensors to perform aggregate queries, each sensor only sends its reading to its parent in the tree. On receiving data from children nodes, the parent aggregates the data and sends the immediate result to its parent, and so on. In this way the sink avoids itself from being flooded by sensor data. [HE04] improves their algorithms by employing multiple sinks, so that the number of messages sent by each sink to the nodes in the data gathering tree is reduced. These works bear similarity with our paper in which we also use intermediate nodes, called *coordinators*, to collect data and reduce communication cost in evaluating aggregate queries. However, we consider *probabilistic* aggregate queries and sensor uncertainty which none of these works does.

The problem of selecting appropriate sensors in a wireless environment for improved accuracy has raised the interest of researchers recently [EFP03, LRZ03, WYPE04]. These works study the accuracy of location tracking in mobile applications. In [EFP03] and [LRZ03], the mutual information between the distribution of an object's location and the predicted location observed by a sensor is used to compute the information gain due to the sensor. The sensor with the highest information gain is selected to reduce the uncertainty of the sensor reading. [WYPE04] developed an entropy-based selection heuristics, and they pointed out that they are computationally more efficient than mutual-information-based methods. It is worth notice that all these studies are only focused on location tracking. Our work encompasses a broader application paradigm, where we consider data uncertainty requirements for common types of continuous queries. To the best of our knowledge, this is the first comprehensive work on *query-based* sensor selection methods.

3. CONTINUOUS PROBABILISTIC QUERIES OVER SENSOR DATA

In this section we first describe the underlying system model. Then we discuss in detail the nature of CPQ and sensor data, and explain how a CPQ is executed in the system.

The wireless sensor system model consists of a *base station* (BS) and a collection of *sensor nodes* distributed in the system environment responsible for capturing the real-time status of their surrounding environment. It is assumed that the system environment is divided into a number of *regions*, each of which consists of a node with high computational capability, called *coordinator node* that manages nodes in the same region. The base station is responsible for communication between the coordinator nodes and the users of the system, i.e., transmitting CPQs to the coordinator nodes and returning results to the users. The base station communicates with the coordinator nodes through a low bandwidth wireless network and may require the relay of other sensor nodes and coordinator nodes. In this paper, we assume that the base station knows the distribution and connections of the coordinator nodes and what sensor data items are represented by each sensor node. Figure 1 illustrates the overall system architecture. Notice that here we assume a two level architecture of the nodes. However, the processing model proposed in this paper can be applied to a multiple-level architecture and the regions managed by different coordinators can overlap.

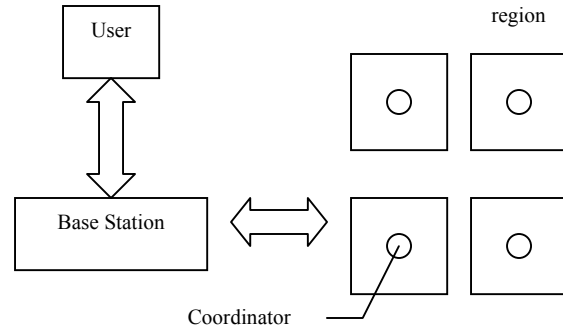


Figure 1. System architecture.

3.1 Query Model

A CPQ is submitted by a user to the sensor network system for the purposes of continuous monitoring and event detection. It is formally defined as follows:

Definition 1. A *Continuous Probabilistic Query (CPQ)* is a probabilistic query repeatedly executed over the time interval $[begin_time, end_time]$ on objects O_1, O_2, \dots, O_n . The answers produced are guaranteed to satisfy the CPQ with some probability specified by the user.

Note that the set of objects to be accessed by a CPQ could be dynamic as each CPQ is referred to regions. When an object moves into a required region of a CPQ, it will be included into the required object set of the query. Here *begin_time* and *end_time* specify the activation period of the CPQ. Once *end_time* of a CPQ has been reached, it will be discarded. It is assumed that a CPQ is defined for specific *regions* [GDG03]. In this paper, we concentrate on *aggregate queries* i.e., those that generate answers based on an aggregate function on O_1, O_2, \dots, O_n . In particular, we study the following three common categories of aggregate queries:

- **MINIMUM (MAXIMUM):** Return the object that contains the minimum (maximum) value among objects O_1, O_2, \dots, O_n with probability guaranteed to be larger than a threshold value P .
- **AVERAGE (SUM):** Return the probability density function of the average value (sum) of objects with the variance of the result not larger than σ_T .
- **RANGE COUNT:** Return the number of data values that fall within a user-specified interval. The probability that the number is the true count has to exceed a threshold value P .

As we can see, answers in a CPQ are augmented with probability values [CKP03]. Further, the probability values have to be constrained by the requirements specified by the user -- which can be viewed as a *quality control metric* over query results. A higher probability or smaller variance indicates a tighter control over answer quality and confidence. The exact form of probabilistic requirements depends on the particular query type. We will examine how these requirements are exploited to select sensors in the next section.

Definition 2. A Continuous Probabilistic Sub-query i , denoted as CPQ_i , is a subquery of CPQ executed during the interval $\{begin_time, end_time\}$. A CPQ_i accesses item O_i in the list of objects specified by CPQ . It returns to CPQ a Gaussian distribution (μ_i, σ_i) of O_i .

In essence, the execution process of a CPQ is as follows: after the base station receives a CPQ, it determines the set of data items required by the CPQ according to the required regions of the query and which coordinator nodes are responsible for generating the required data items. The base station then breaks down the CPQ into sub-queries $\{CPQ_1, CPQ_2, \dots, CPQ_n\}$. Each sub-query CPQ_i is then sent to the coordinator node, which is responsible for reading O_i and generating a Gaussian distribution for the reading of O_i to describe the distribution of O_i . Each coordinator then sends its results back to the base station; it then computes and sends back the final result to the user. Figure 2 illustrates an example of a CPQ executing on objects O_1 and O_4 under our system model.

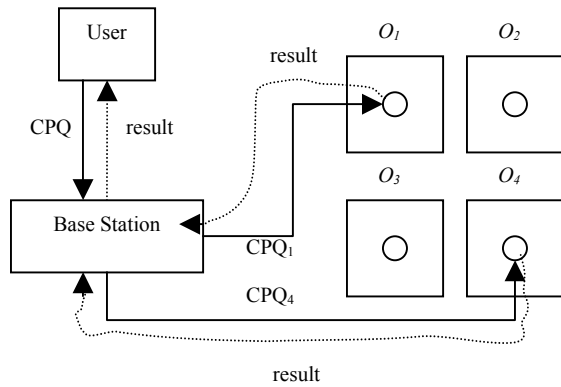


Figure 2. The CPQ submitted by the user is broken down into two sub-queries, CPQ_1 and CPQ_4 , which access regions O_1 and O_4 , respectively. The results from coordinators for O_1 and O_4 are sent to the Base Station, which subsequently returns the final result to the user.

3.2 Sensor Data

It is assumed that the sampled values from a sensor follow a *continuous* function of the system environment. Typical examples include temperature and pressure values. Each sensor has a pre-defined sampling period defined based on the maximum rate of changes in the status of the sampling signal and the accuracy requirements of the queries to be accessed to it. Thus, a stream of sensor values is generated from a sensor representing the system status as a function of time as shown in Figure 3. Normally, if the rate of change is lower, a larger sampling period may be adopted. Of course, a smaller sampling period provides a better modeling on the continuous status of the system environment. However, the sampling cost and the number of data values generated will be higher. The consequence is that more energy is consumed at the node.

Once a sensor data value is generated, it is associated with two time-stamps, the lower time stamp (lts) and upper time stamp (uts), to indicate its validity interval. A data value is invalid at the sampling time of the next value. The time-stamps can be used for ensuring temporal consistency in query execution [LP04, SBLC03]

such that all the data values are valid at the same time interval. The synchronization of the clocks at different sensor nodes for generating the time-stamps can be done by the coordinator node of the region.

The generated sensor data values from a sensor node may contain error due to noises from the surrounding environment. In Figure 3, for example, the sampled value at time t_4 significantly deviates from the actual status (which can be determined based on the assumed maximum rate of changes of the sensor readings). The reliability of a sampled value can be checked with the previously sampled values. It is expected that the next sampled value should not differ significantly from the current value.

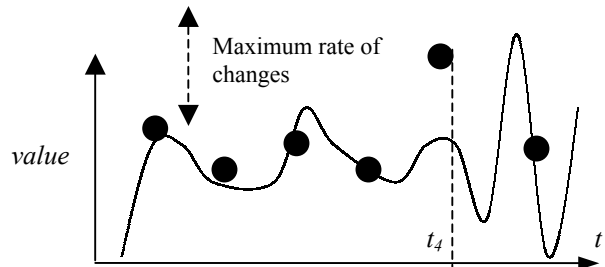


Figure 3. Continuous data values from a sensor node.

3.3 Coordinator Node

Now let us understand the model of a coordinator node in more details. As mentioned before, a coordinator node is a computationally powerful node responsible for managing the sensor nodes required by the queries, as well as processing queries that access the nodes in the same region. As shown in Figure 4, the coordinator node collects readings from individual sensors (possibly with weaker computing capability). It then computes the average value of these sensors, and returns the statistical information (e.g., Gaussian distribution) of these sensor values. The mean value and statistical information will be passed to the base station for generating query results. Notice also in Figure 4 that *not* all the sensors are involved in sending their values to the coordinator. This is because reading data values from all sensors can be expensive. We will examine how a coordinator chooses sensors in more detail in next section.

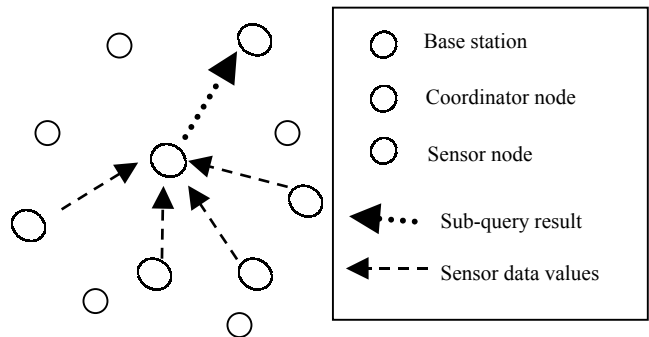


Figure 4. Coordinator Model.

To conclude, the processing of a CPQ is divided into two levels: (1) Aggregation of multiple sensors responsible for generating a value

for a data item required by the query at the coordinator, and (2) Aggregation of the data items required by the query to generate the query results to be performed at the base station. Since CPQ is a continuous query, aggregation operations will be performed whenever a new set of sensor data versions are generated. The aggregation at the coordinator node will follow the time-stamps of the data values of different sensor data streams such that they are *relatively consistent* with each other as shown in Figure 5 [LP04]. Although the sampling periods of the sensor nodes within the same region may be the same (i.e., the sensor values have the same validity period length), the sensor data values from different streams may not be synchronized with each other since they may be activated at different time points as shown in Figure 5. The results of a CPQ are then a sequence of values as a function of time.

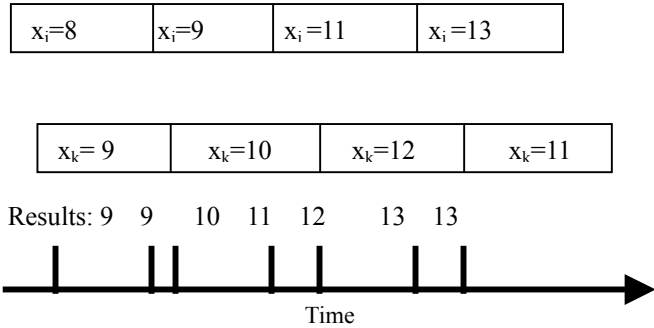


Figure 5. Accessing sensor data value with relative consistency requirement. An aggregate function, *max*, is performed on two sensor streams x_i and x_k . The results are obtained from the maximum of the two streams for the same time points. They are presented as a function of time.

4. ACCESSING SENSOR DATA BY A CPQ

Accessing more sensor nodes can improve the reliability of query results. However, it is also important to minimize the aggregation workload. Our goal is to meet the accuracy (probabilistic requirement) of a query using minimum number of sensor nodes for generating the value of a data item required by a query. The problem we want to solve is to determine the set of sensor nodes to participate in sampling for aggregation of the values at the coordinator node. Since different types of probabilistic queries have different forms, the derivation of the probabilistic query result (in the form of probability density function, or *pdf*) can be different. This in turn affects how many sensors are selected to report data. We propose an approach which can be summarized in three steps to be performed:

1. Each coordinator involved in a query computes the mean value of the data item.
2. The base station derives the maximum allowed variance of the sample data for meeting the query accuracy requirements.
3. Each coordinator then determines the sample size and the sensor nodes to be used.

In the rest of the section, we will discuss each step in more detail.

4.1 Computing the Mean of Data Values

For each data item O_i required by a sub-query CPQ_i , the

coordinator node identifies the set of sensor nodes S_i which are responsible for generating value for O_i . Then it sends out data request messages to all these sensor nodes. Each sensor node responds to the request message by returning its latest sampled data value of O_i to the coordinator. The received sensor data values from each sensor node are put into a buffer first. When the coordinator has received all the data values from the sensor nodes, it calculates the mean value of the data values. The calculation may be performed when the waiting time for sampled data values has been expired, even though not all the sensor nodes have returned their values -- those nodes may be out of operations or disconnected due to communication error.

To improve the accuracy in calculating the mean value, each node may send multiple sensor values which cover the period from (*current time - a time interval*) to *current time*. Then, the mean value calculation is performed in two steps: (1) the mean value for the set of data values from a sensor node, and (2) the mean value over all sensor nodes. If the variance of the values from a sensor node is too high, (e.g., higher than a pre-defined threshold), it is assumed that the node is either currently located at the high-noise environment or the node is currently operating in an abnormal state. Then the node will be marked as abnormal and the coordinator will not consider the node for further processing for a period of time even when it is required later.

4.2 Deriving Maximum Allowed Variance

In this step, the base station analyzes the distribution of results for different types of probabilistic queries. Based on the distribution of the probabilistic results from the coordinator nodes, the probabilistic requirements of the queries, and the mean values of the sensor data obtained in Step 1 (Section 4.1), it derives the *variance* σ of the sampled data being queried.

The impact of errors in sampled data values on the query result depends on the query type. Here we consider three query types: (1) Range count queries, (2) MAX and MIN, (3) AVG and SUM. The data being queried are aggregated from multiple sensors from the same regions by the coordinator nodes. Thus it is reasonable to assume that the data values follow normal distributions with specific means and variances. In essence, the sub-query CPQ_i executed at each coordinator returns a normal distribution of its sensor reading to the base station.

The rest of this section describes, for each query, how the maximum variance allowed for an object comes into play. We then derive an algorithm to compute the variance.

(1) MAXIMUM and MINIMUM

One important observation about MAXIMUM and MINIMUM queries is: if the variance of the sampled data values is smaller, the maximum and the minimum will be more distinct. In the example of two data values below, the probability of O_1 being the maximum data object, i.e. the value of O_1 is larger than that of O_2 , is:

$$p_1 = \int_{-\infty}^{+\infty} f_1(s) \cdot \left(\int_{-\infty}^s f_2(t) dt \right) ds.$$

It can be seen from Figure 6 below, if the variance is smaller, p_1 will be larger. It is consistent with the fact that O_1 is more likely to be the maximum.

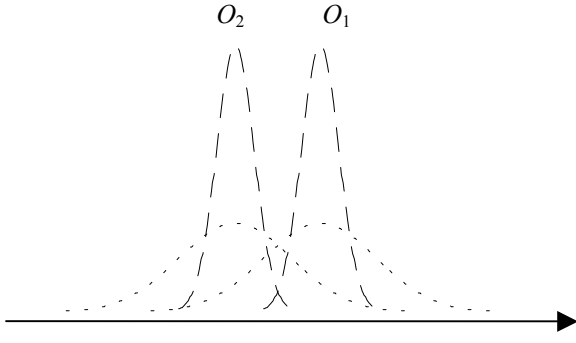


Figure 6. MAXIMUM and MINIMUM queries.

For the case of multiple data values, the probability of O_i being the maximum, i.e. its value is larger than any other data objects concerned, is:

$$p_i = \int_{-\infty}^{+\infty} f_i(s) \cdot \left(\prod_{j \neq i} \int_{-\infty}^s f_j(t) dt \right) ds,$$

where the probability density functions are:

$$f_i(s) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(s-\mu_i)^2}{2\sigma_i^2}} \quad \text{and} \quad f_j(t) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(t-\mu_j)^2}{2\sigma_j^2}}.$$

Let σ_{req} 's be the required variances of objects being monitored, for which the probability of each object holding the maximum value is guaranteed to be larger than $P\%$. The following algorithm finds σ_{req} 's for MAX query.

Algorithm 1: finding the σ 's to meet the accuracy requirement of query type (2) MAX

1. Set the σ_{req} 's to be σ of a single sensor, which is the maximum variance.
2. Find i_{max} , the index of the maximum p_i
3. Find j_{max} , the index of the maximum $\frac{\partial}{\partial \sigma_j} p_{i_{max}}(\sigma_1, \sigma_2, \dots, \sigma_n)$ i.e., the sensor that has the greatest impact to $p_{i_{max}}$.
4. Adjust variance requirement of the j_{max}^{th} sensor: $\sigma_{j_{max}} = \sigma_{j_{max}} - \Delta\sigma$
5. Repeat 2 to 4 until $p_{i_{max}}(\sigma_1, \sigma_2, \dots, \sigma_n) \geq P\%$
6. Return $\sigma_1, \sigma_2, \dots, \sigma_n$ as σ_{req} 's

Note that Algorithm 1 is a greedy algorithm. Each time it adjusts the sensor's variance requirement that has the greatest impact on the accuracy of the results.

(2) AVERAGE and SUM

The pdf of the resultant SUM is the convolution of the pdf of the addends. We first define the convolution operator \otimes as: $(f \otimes g)(z) := \int_{-\infty}^{+\infty} f(s)g(z-s)ds$. Then the pdf of the sum of

the values of $O_{k_1}, O_{k_2}, \dots, O_{k_m}$, whose pdfs are $f_{k_1}(t), f_{k_2}(t), \dots, f_{k_m}(t)$, is:

$$f_S = f_{k_1} \otimes f_{k_2} \otimes \dots \otimes f_{k_m}$$

It is a parameterized function with $\sigma_{k_1}, \sigma_{k_2}, \dots, \sigma_{k_m}$ as

parameters, since $f_{k_j}(t) = \frac{1}{\sqrt{2\pi}\sigma_{k_j}} e^{-\frac{(t-\mu_{k_j})^2}{2\sigma_{k_j}^2}}$. According to the

probability theory, the sum of a finite number of normal-distributed random variables still follows normal distribution. More specifically, for our case, the sum follows

normal distribution $N(\sum_{j=1}^m \mu_{k_j}, \sum_{j=1}^m \sigma_{k_j}^2)$. We find

$\sigma_{k_1}, \sigma_{k_2}, \dots, \sigma_{k_m}$ to meet the requirement that

$\sigma_S = (\sum_{j=1}^m \sigma_{k_j}^2)^{1/2}$ is smaller than a given threshold σ_T . Please note

that AVG is SUM divided by m . Thus it follows the same rationale except that its distribution is given

by $N(\frac{1}{m} \sum_{j=1}^m \mu_{k_j}, \frac{1}{m^2} \sum_{j=1}^m \sigma_{k_j}^2)$. Algorithm 2 illustrates how σ 's

are found for SUM and AVERAGE.

Algorithm 2: finding the σ 's to meet the accuracy requirement of query type (3) SUM

1. Set the σ_{req} 's to be σ of a single sensor, which is the maximum variance.
2. Find k_{max} , the index of the maximum $\frac{\partial}{\partial \sigma_k} \sigma_S = \frac{1}{2} (\sum_{j=1}^m \sigma_{k_j}^2)^{-1/2} \cdot 2\sigma_k = \sigma_k (\sum_{j=1}^m \sigma_{k_j}^2)^{-1/2}$ i.e., the sensor that has the greatest impact to σ_S .
3. Adjust variance requirement of the k_{max}^{th} sensor: $\sigma_{k_{max}} = \sigma_{k_{max}} - \Delta\sigma$
4. Repeat 2 to 4 until $\sigma_S \leq \sigma_T$
5. Return $\sigma_1, \sigma_2, \dots, \sigma_n$ as σ_{req} 's

(3) Range Count

A range count query Q specifies an upper query bound (ub_Q) and a lower query bound (lb_Q). An object will be counted if it is within the range. The result is a count, i.e., counting the number of objects within the range. For a probabilistic range count query, the result needs to provide the confidence of the counted value, i.e., the value whose probability of falling inside the specified range is not smaller than a given value, e.g. 95%. From Figure 7, we can see that if the value of the sampled data value being queried is near the query bound (i.e., O_j), a smaller error needs to be provided from the data value. Otherwise, it cannot determine with high confidence whether the sample value is within or out of the query bound, and consequently leads to a low confidence in the count value. To minimize the sampled error, more samples may be collected from the sensor nodes. On the other hand, if it is away from the bounds (i.e., O_2), a larger error will be accepted as it will not have any significant impact on the results.

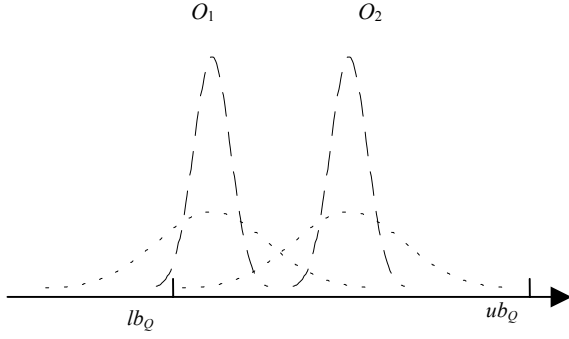


Figure 7. Probabilistic Range Count Query.

To determine σ_i for O_i in range count queries, we need to understand first the impact of σ_i on the result of a *probabilistic range query* (PRQ), which computes the probability that an object value falls within the specified range. Suppose the mean of the value of O_i is μ_i , and the lower bound and upper bound of the range query being discussed are lb_Q and ub_Q respectively. The probability that O_i satisfies the PRQ, p_i , is then given by:

$$\frac{1}{\sqrt{2\pi}\sigma_i} \int_{lb_Q}^{ub_Q} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} dx \geq P\%$$

Notice that the value of p_i is a function of variance, i.e., the smaller are the variance is, the more accurate is the probability value.

There are two types of results for a range count query:

- (1) *Deterministic*: Count all data objects with at least 95% chance (a confidence requirement example) of being inside the range.
- (2) *Probabilistic*: Find the distribution of the result (called COUNT) to a range count query:

COUNT	1	2	3	4	5
Prob.	0.1	0.2	0.5	0.15	0.05

We may require the distribution to be more concentrated, e.g. 95% of the probability is concentrated on the top 2 high probability COUNT values.

We focus on the results given in probabilistic forms. We now show how to get this distribution. Assuming that there are n data objects: O_1, O_2, \dots, O_n . The standard variances of the values of these data object are $\sigma_1, \sigma_2, \dots, \sigma_n$. Suppose the probabilities are $p_1(\sigma_1), p_2(\sigma_2), \dots, p_n(\sigma_n)$, respectively. We can get the distribution of the random variable COUNT by examining all the combinations, i.e.:

$$\begin{aligned}
P(\text{COUNT}=0) &= [1-p_1(\sigma_1)] \cdot [1-p_2(\sigma_2)] \cdot \dots \cdot [1-p_n(\sigma_n)] \\
P(\text{COUNT}=1) &= p_1(\sigma_1)[(1-p_2(\sigma_2)) \cdot \dots \cdot (1-p_n(\sigma_n))] + p_2(\sigma_2)[(1-p_1(\sigma_1)) \cdot (1-p_3(\sigma_3)) \cdot \dots \cdot (1-p_n(\sigma_n))] + \dots + p_n(\sigma_n)[(1-p_1(\sigma_1)) \cdot \dots \cdot (1-p_{n-1}(\sigma_{n-1}))] \\
P(\text{COUNT}=2) &= p_1(\sigma_1) p_2(\sigma_2) [(1-p_3(\sigma_3)) \cdot \dots \cdot (1-p_n(\sigma_n))] + p_1(\sigma_1) p_3(\sigma_3) [(1-p_2(\sigma_2)) \cdot \dots \cdot (1-p_n(\sigma_n))] + \dots + p_1(\sigma_1) p_n(\sigma_n) [(1-p_2(\sigma_2)) \cdot \dots \cdot (1-p_{n-1}(\sigma_{n-1}))] + \dots
\end{aligned}$$

$$\begin{aligned}
& p_2(\sigma_2) p_3(\sigma_3) [(1-p_1(\sigma_1)) \cdot \dots \cdot (1-p_n(\sigma_n))] + p_2(\sigma_2) p_4(\sigma_4) [(1-p_1(\sigma_1)) \cdot \dots \cdot (1-p_n(\sigma_n))] + \dots + \\
& p_2(\sigma_2) p_n(\sigma_n) [(1-p_1(\sigma_1)) \cdot \dots \cdot (1-p_{n-1}(\sigma_{n-1}))] + \\
& p_3(\sigma_3) p_4(\sigma_4) [(1-p_1(\sigma_1)) \cdot \dots \cdot (1-p_n(\sigma_n))] + p_3(\sigma_3) p_5(\sigma_5) [(1-p_1(\sigma_1)) \cdot \dots \cdot (1-p_n(\sigma_n))] + \dots + \\
& p_3(\sigma_3) p_n(\sigma_n) [(1-p_1(\sigma_1)) \cdot \dots \cdot (1-p_{n-1}(\sigma_{n-1}))] + \\
& \dots \\
& p_n(\sigma_n) p_{n-1}(\sigma_{n-1}) [(1-p_1(\sigma_1)) \cdot \dots \cdot (1-p_{n-2}(\sigma_{n-2}))] \\
& \dots \\
P(\text{COUNT}=n) &= p_1(\sigma_1) \cdot p_2(\sigma_2) \cdot \dots \cdot p_n(\sigma_n)
\end{aligned}$$

From the requirement on this distribution, e.g. 95% of the probability is concentrated on the top 2 highest probability COUNT values, we need to find a way to derive $\sigma_1, \sigma_2, \dots, \sigma_n$ based on the above distribution.

Algorithm 3: finding the σ 's to meet the accuracy requirement of probabilistic RANGE COUNT query such that a given percentage $P\%$ of probability is concentrated on the top k COUNT values.

1. Set the σ_{req} 's to be σ of a single sensor, which is the maximum variance.
2. Find i_1, i_2, \dots, i_k of the top $P(\text{COUNT}=i)$ values.
3. Find j_{max} , the index of the maximum $\frac{\partial}{\partial \sigma_j} \sum_{m=1}^k P(\text{COUNT} = i_m)$ i.e., the sensor that has the greatest impact to the sum.
4. Adjust variance requirement of the j_{max} th sensor: $\sigma_{j_{max}} = \sigma_{j_{max}} - \Delta\sigma$
5. Repeat 2 to 4 until $\sum_{m=1}^k P(\text{COUNT} = i_m) \geq P\%$
6. Return $\sigma_1, \sigma_2, \dots, \sigma_n$ as σ_{req} 's

4.3 Determining Sample Size and the Set of Sensors

In this final step, the coordinator node determines the sample size and the set of sensor nodes to be sampled to meet the confidence requirement of data being queried, i.e. variance σ_{req} .

A correlation-based approach is adopted. It is based on the correlation coefficient of the historical value of individual sensor node and their average. Suppose the values of all sensor nodes in an area are $s_i, i = 1, 2, \dots, n$. Their average is denoted

$$as E(s) = \frac{1}{n} \sum_{i=1}^n s_i. Some of the n sensor nodes are requested to$$

send their values periodically to the coordinator node for aggregation to get the average value of that area to approximate $E(s)$. The number is minimized to reduce cost and power consumption. With a far larger period, the coordinator node pulls values from all sensor nodes to get the exact $E(s)$, and then calculate the correlation coefficient between $E(s)$ and the value of each sensor. Based on the correlation information, we reselect the sensors which are correlated higher with $E(s)$ among others and

transmit its value for approximation of $E(s)$ during the coming long period. Of course, the sample size should be large enough to meet the confidence requirement. It can be determined based on the historical statistics of sensor values. Note that the selection process should prefer sensor nodes that are operating in a normal situation.

We first decide how many sensors are requested to participate in the aggregation to get the average value, i.e. the sample size. Suppose the sample size is n_s . The approximate mean

$$\text{value } \bar{S} = \frac{1}{n_s} \sum_{i=1}^{n_s} s_{k_i}, \text{ where } 1 \leq k_i \leq n \text{ and } k_i \neq k_j \text{ for all } i \neq j. \text{ We}$$

know that if all s_{k_i} follow an identical distribution $N(\mu, \sigma^2)$,

then \bar{S} follows the normal distribution $N(\mu, \sigma^2/n_s)$, where $\mu =$

$$E(s) \text{ and } \sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (s_i - E(s))^2}. \text{ To satisfy the accuracy}$$

requirement, we need to choose a n_s value satisfying the constraint $\sigma/\sqrt{n_s} \leq \sigma_{req}$. So we set the sample size to be:

$$n_s = \left\lceil \sigma^2 / \sigma_{req}^2 \right\rceil.$$

To minimize the effect of erroneous sensor readings whose values deviate significantly from the majority, we decide to choose the top n_s sensors whose values are highly correlated with $E(s)$. The degree of correlation is calculated based on the correlation coefficient. For a sensor s_i and $E(s)$, we first calculate their covariance:

$$\text{cov}(s_i, E(s)) = E((s_i - E(s))(E(s) - E(E(s))))).$$

Please note that, s_i and $E(s)$ are both a stream of values from successive pulls, and $E(s_i)$ is the mean of the stream of values of s_i ; while $E(E(s))$ is the mean of the stream of values of $E(s)$. With the covariance between s_i and $E(s)$ in hand, we can proceed to calculate the correlation coefficient between them:

$$\rho_i = \frac{\text{cov}(s_i, E(s))}{\sqrt{D s_i} \sqrt{D(E(s))}}.$$

We sort ρ_i , $i = 1$ to n in descending order, and then choose the top n_s correlated sensors to be sampled.

4.4 Other Issues of the Sensor Selection Process

A natural question to ask about the 3-step selection process is: when do we need to execute this process again? This is an important issue because queries are continuous, and if there is any change in the state of sensor reading (e.g., aggregate value of a sensor within a region), the sensor selection process may have to be re-executed in order to select the right set of sensors, so that the result of the continuous query is guaranteed to be correct throughout its lifetime. A simple solution to this question is to execute the selection process periodically. The period should be fine enough compared with the lifetime of a query, but should not be too small as this may cause unnecessary burden to the base station and the coordinator. Another more elegant solution is to enforce the base station to track the reading value sent by each coordinator. If the newest reading deviates too much from the (old)

value used to execute the selection process, the base station can choose to restart the selection process.

Another situation where the selection process may be re-evaluated is when the set of objects being monitored by the CPQ is changed. For example, when an object moves into a required region of a CPQ, it can be included into the required object set of the query, possibly triggering the selection process.

Finally, if an object is involved in multiple queries, each query can have different requirement of σ_i for the object. In this case we simply use the minimum σ_i derived from the requirements of all queries concerned.

5. FURTHER DISCUSSIONS

We identify a number of interesting issues that are worth further investigation. Recall that our algorithm distinguishes a sensor that reports unusual values from others by statistical techniques. Notice that this ‘‘abnormal’’ sensor may be used again in the next selection phase. In reality, this sensor may be faulty and should not even be considered by the coordinator in the future. Nevertheless, in some cases a sensor reporting unusual behaviors need not be erroneous. For example, if the corner of a room is on fire, the temperature sensor located at the corner may reveal unusually high temperature values compared with other sensors. It may not be wise for the coordinator to ignore this sensor. An interesting question is: how can we distinguish ‘‘bad’’ sensors from those that genuinely detect unusual behaviors?

In this paper, we implicitly assume that a coordinator is assigned to a fixed number of redundant sensors that are in the same operating range. That is to say, under normal operation, the redundant sensor readings are assumed that they do not deviate much from each other. Also, the coordinator is not overloaded by the sensors. In scenarios where sensors can be mobile (e.g., sensors embedded in moving objects), we may need a more flexible scheme. For example, a coordinator may get overloaded, so we may need to share the workload among the coordinators. In particular, it may be necessary to develop a scheme where configurations of a coordinator (e.g., which sensor it is assigned to monitor) can be changed during operation.

The base station in our model communicates directly with coordinators. The rationale for this design is to relieve the base station from having the need to communicate directly with all sensors -- the coordinators provide a level of aggregation so that the communication cost between the base station and sensors is significantly reduced. In a large geographical area, there can be many coordinators, so that their communication with the base station may also be a bottleneck. To solve this problem, we can build another level of coordinators, each of which communicates with a group of coordinators. Ultimately, a system involving multiple levels of coordinators and sensors can be developed. The question is then what criteria should be used to define the structure of the network connecting the coordinators? Also, how can a CPQ be expressed as sub-queries in such a system?

We assume that each sensor reports the actual value it obtains from the environment. We do not consider any error inherent in a sensor such as its sampling and measurement error. If the issues of uncertainty of a sensor are taken into account, the accuracy of a query can be improved through the evaluation of probabilistic queries [CKP03]. In particular, the uncertainty of a sensor can be reduced if it is sampled more frequently, resulting in higher

reliability in query results. With the improved reliability in each sensor reading, we may be able to reduce the number of sensors required by the algorithms presented in this paper. A better scheme may be changing our proposed algorithm to take into account uncertainty in each sensor, and by controlling sampling rate, achieves reliable results with fewer redundant sensors.

Finally, we only studied some common types of aggregate queries in this paper. We may examine whether our proposed scheme can work for *non-aggregate queries* that do not require aggregate operators over objects [CKP03]. A typical example is a range query where the probability of each object falling within the range is independent of other objects. It is also a challenging problem to study whether our scheme can be extended to handle *general SQL-type queries*, which involve both aggregate and non-aggregate operators.

6. CONCLUSIONS

With prices of sensors continuously dropping, we expect that more applications will deploy large sensor networks for monitoring purposes. In this paper, we exploit the availability of low-cost sensors and develop a comprehensive scheme that selects appropriate sensors to provide reliable query results. Particularly, we develop a framework where coordinators manage, aggregate redundant sensors, and report the results to the base station. We also illustrate that different types of queries have different accuracy requirements. Under our proposed framework, we devise a probabilistic approach to select sensors for common aggregate query types. Our next step is to test our model in real-time simulation experiments, and optimize the performance of our algorithms.

8. ACKNOWLEDGEMENT

The described in this paper was partially supported by a grant from the Research Grant Council of Hong Kong Special Administration Region, China [Project No. CityU 1076/02E].

7. REFERENCES

- [CKP03] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, June 2003.
- [EN03] E. Elnahrawy and B. Nath. Cleaning and Querying Noisy Sensors. In *ACM WSNA'03*, September 2003, San Diego, California.
- [EFP03] E. Ertin, J. Fisher, and L. Potter. Maximum mutual information principle for dynamic sensor query problems. In *Proc. IPSN'03, Palo Alto, CA*, April 2003.
- [HE04] H. Dubois-Ferriere and D. Estrin. Efficient and Practical Query Scoping in Sensor Networks. *CENS Technical Report #39*, April 2004.
- [MFHH02] S. Madden, M. J. Franklin, J. Hellerstein, and W. Hong. Tiny aggregate queries in ad-hoc sensor networks. In: *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, USA (2002).
- [K03] V. Kumar. Sensor: The Atomic Computing Particle. In *SIGMOD Record*, Vol. 32, No. 4, December 2003.
- [LP04] K. Y. Lam and H. C. W. Pang, Correct Execution of Continuous Monitoring Queries in Wireless Sensor Systems. In *Proceedings of the Second International Workshop on Mobile Distributed Computing (MDC'2004)*, Tokyo, Japan, March 2004.
- [LRZ03] J. Liu, J. Reich, and F. Zhao. Collaborative in-network processing for target tracking. In *EURASIP JASP: Special Issues on Sensor Networks*, 2003(4):378-391, March 2003.
- [N04] The National Institute of Standards and Technology. Wireless Ad Hoc Networks: Smart Sensor Networks. URL: http://w3.antd.nist.gov/wahn_ssn.shtml
- [NN04] D. Niculescu and B. Nath. Error characteristics of adhoc positioning systems. In *Proceedings of the ACM Mobihoc 2004*, Tokyo, Japan, May 2004.
- [SBLC03] M. A. Sharaf, J. Beaver, A. Labrinidis, P. Chrysanthis. TiNA: A Scheme for Temporal Coherency-Aware in-Network Aggregation. In *Proceedings of 2003 International Workshop in Mobile Data Engineering*.
- [BMEP03] V. Bychkovskiy, S. Megeriam, D. Estrin. and M. Potkonjak. A collaborative approach to in-place sensor calibration. In *Proceedings of IPSN'03* (2003).
- [WYPE04] H. Wang, K. Yao, G. Pottie, and D. Estrin. Entropy-based Sensor Selection Heuristic for Localization. In *3rd International Workshop on Information Processing in Sensor Networks (IPSN'04)*, April 2004.
- [WYYE04] H. Wang, L. Yip, K. Yao and D. Estrin. Lower Bounds of Localization Uncertainty in Sensor Networks. In *Proceedings of IEEE International Conference on Acoustics, Speech*, May 2004.