

Scalable and Robust Aggregation Techniques for Extracting Statistical Information in Sensor Networks

Hongbo Jiang and Shudong Jin

Division of Computer Science, EECS, Case Western Reserve University
Cleveland, OH 44106, USA
{hongbo.jiang,shudong.jin}@case.edu

Abstract

Wireless sensor networks have stringent constraints on system resources and data aggregation techniques are critically important. However, accurate data aggregation is difficult due to the variation of sensor readings and due to the frequent communication failures. To address these difficulties, we propose a scalable and robust data aggregation algorithm. The novelty of our work includes two aspects. First, our algorithm exploits the mixture model and the Expectation Maximization (EM) algorithm for parameter estimation. Hence, it captures the effects of aggregation over different scales while keeping the communication cost low. Second, our algorithm exploits loss-tolerant multi-path routing schemes. Hence, it obtains accurate statistical information even in the presence of high link and node failure rates. We demonstrate that our techniques reduce communication cost while retaining the precious statistical information otherwise neglected by other aggregation techniques. Our evaluation shows the proposed techniques are robust against link and node failures, and perform consistently well.

1. Introduction

Wireless sensor networks have a broad range of applications, such as battlefield surveillance, environmental monitoring, and disaster relief. A sensor network consists of a set of autonomous sensor nodes which spontaneously create impromptu communication links, and then collectively perform tasks without help from any central servers. Often an individual sensor node is an inexpensive wireless device with limited computational power and energy supply. To prolong the lifetime of a sensor network, one of the key problems is *energy-efficient* data extraction. That is, how can a sink (also the querying node) obtain the sensor readings with a low energy consumption? To that end, a number of network support techniques [1, 2, 3, 12, 14, 17] have been proposed.

Two key points should be elaborated with respect to these studies. First, exact data extraction is often considered im-

practical when packet losses are eminent and the sensor readings are variable. To track the statistical properties of sensor data in many applications, approximate estimates are often sufficient, and sometimes are desirable if energy conservation is taken into consideration. For example, if the sensors report temperature readings, it is not necessary for each sensor to report its entire data stream in full fidelity. Recent approximation methods [1, 2, 3] have already vitalized this trend. The common characteristic of these studies is to reduce the number of sensor readings and radio transmissions. Second, it is desirable to obtain the overall data distribution and the correlation information across the entire sensor network. This capability provides a much broader and more robust summarization of the behavior of the entire network [9], and is important to supporting more complex queries. Unfortunately, previous studies have not provided any specific techniques to enable this. To the best of our knowledge, only [13] partially provides this capability.

This paper addresses both points. We present a scalable and robust aggregation technique based on statistical information extraction for arbitrary sensor networks. We aim to answer the queries efficiently and accurately with minimum communication cost. Approximation solutions are considered. The solutions should have limited communication overhead in large-scale sensor networks, and they should be robust against both link and node failures. In our distributed aggregation algorithm, we apply the novel idea of utilizing the mixture model to approximate the exact data distribution. By doing so the system can have excellent scalability. Our methods also exploit a loss-tolerant framework with the idea of multi-path routing. By doing so, the system is more robust. Via simulation study we demonstrate that our aggregation techniques can provide statistical information accurately, that they have bounded communication cost, and that they are robust against link and node failures.

The rest of the paper is organized as follows. In Section 2 we will describe related work on data extraction in sensor networks. In Section 3, we describe our data aggregation algorithm and our methods for estimating distribution parameters. Section 4 briefly analyzes the complexity and

discusses the energy consumption of the algorithm. Section 5 provides a performance comparison of different aggregation techniques. Section 6 concludes the paper.

2. Related Work

Energy conservation is crucial to the prolonged lifetime of a sensor network. Since radio operation is by far the largest factor of energy consumption in sensor nodes, many approaches have been explored to minimize the number of transmissions. Broadly, they can be classified into two categories. The first category focuses on designing better routing protocols and delivery algorithms. For example, Trigoni et al. [14] proposed a scheduling algorithm to calculate energy and delay carefully for energy-efficient data dissemination. Considine et al. [1] presented a fault-tolerant routing technique using multi-path routing. It uses multicast trees and takes advantage of the broadcast property of omnidirectional antennas. Park et al. [10] proposed a scalable framework for reliable downstream data delivery from the sink to the sensors that is specifically designed to leverage the characteristics of a sensor network, while achieving reliability in an efficient manner. However, one major drawback of these approaches is they try to collect precise information, and thus incur high traffic and high energy consumption.

The second category focuses on selecting more representative and concise sensor data for delivery. For example, Sadagopan et al. [12] formulated the problem of maximizing data collection from an energy constrained wireless sensor network as a linear programming problem and provided an iterative algorithm. Deligiannakis et al. [2] used a compression technique to process historical information by removing correlated and redundant measurements. Probabilistic query methods [3] exploited a statistical model with probabilistic confidence level to reduce data acquisition. However, these approaches often require global information and they have limited scalability.

In-network data aggregation is a distributed technique that exploits both energy-efficient routing and data summarization. Several in-network aggregation systems have been designed, with particular attention to aggregate query processing. A straightforward method is to compute aggregates such as AVERAGE, SUM and COUNT over a routing tree, minimizing both the number of messages as well as the size of the messages. For example, Cougar [16], TinyDB and TAG [7] have used this method. This method has to overcome two limitations. First, a sensor network has often moderate to high node and link failure rates, due to environmental interferences, collisions, and the low signal-to-noise ratio. They may cause the loss of a large amount of information. Failures are inevitable in a routing tree-based aggregation. If they happen at a location close to the sink, the impact on the resulting aggregate can be significant. Second, aggregation measures such as AVERAGE, SUM and COUNT are not sufficient in some applications, and the distribution of sensor

readings is necessary [9]. If an estimate of the data distribution is available at the sink, users can pose more complex queries and perform more sophisticated analysis.

To overcome the first limitation, in-network aggregation must be robust against link failures and packet losses. Retransmission-based approaches are feasible in providing reliability in wired networks such as the Internet. However, they are too expensive in wireless sensor networks. Hence, multi-path routing based solutions were proposed [1, 4, 5, 7, 8]. For example, the TAG system [7] takes such an approach to obtaining a fault-tolerant solution. The approach is efficient for simple MIN and MAX aggregations, but it does not work well for duplicate-sensitive aggregates. Considine et al [1] extended this approach and adopted the duplicate-insensitive sketches for approximating COUNT to handle SUM. Nath et al. [8] proposed synopsis diffusion that compactly summarizes intermediate in-network aggregation. Synopsis diffusion creates rings of nodes centered around the sink, and then the aggregate values are continually routed towards the sink.

To overcome the second limitation, several studies have focused on providing statistical summarization. For example, Deshpande et al. [3] proposed a statistical model to enrich interactive sensor querying. They integrated a database system with a correlation-aware probabilistic model. It reduces the number of expensive sensor readings and radio transmissions that the networks must perform. Prior to that, Yao and Gehrke [17] used the Gaussian ADT to model the uncertainty as a continuous probability distribution function over possible measurement values. On the contrary, Shrivastava et al. [13] presented approximation algorithms to estimate statistical values such as median and quantiles. They used a data structure called *quantile digest* to preserve information about high frequency values while compressing information about low frequency ones. They have also shown the message cost and errors are bounded.

3. Aggregation Algorithms

As our objective is to obtain the statistical information of the sensor data, it suffices if the aggregation algorithms return the probability distribution of the data. To that end we exploit general mixture model techniques. In this section we present the theoretical foundation, describe the process of aggregation, and formulate and solve the problem of estimating distribution parameters.

3.1. Network Models

Consider a sensor network where the link between a pair of sensor nodes is lossy. Packets can be corrupted or dropped in harsh environments where link (and route) failures are possible. The routing algorithm (and protocol) decides how to forward data towards the sink. With multi-path routing, each node can have multiple predecessors and successors in the

routing graph. Each node aggregates the results received from its predecessors and the value from its own reading, and then sends the result to one or more of its successors. We ignore the details on how to generate the routing graph, but point out that our aggregation algorithm (which is the focus of this paper) can work with any multi-path routing scheme.

3.2. Theoretical Approximation

Consider a sequence of samples, sensor readings. Let $f(X)$ denote the continuous probability model for sample X . Its probability density function is $f(x)$. Theoretically, an infinite dimension mixture model can asymptotically approach any probability density function [11]. That is, for any continuous probability density function, if we properly select the parameters of the base distributions in the mixture model, then we have,

$$f(x) = \lim_{N \rightarrow \infty} \sum_{i=1}^N \alpha_i B_i(x; \theta_i)$$

where $B_i(x; \theta_i)$ is the i -th base distribution with parameter θ_i , and $\sum_{i=1}^N \alpha_i = 1$. In the rest of paper, we use Θ to denote all the parameters on the right side (all θ_i and all α_i). Hence, this leads to an aggregation technique: the sensor nodes need only to transmit packets that contain the distribution parameters instead of the individual values. If we can choose the parameters carefully, a good approximation is attainable. Furthermore, since the number of parameters determines the message size, we can adjust it to trade-off the error rate and the communication cost. We design our aggregation algorithm based on this theoretical approximation.

3.3. Aggregation Process

The general process of our aggregation algorithm is as follows. The aggregation starts with remote nodes towards the sink. A remote node may first send packets to its successors using multi-path routing. The routes to the sink should be established before the actual aggregation starts. Upon receiving packets from other nodes, intermediate nodes will aggregate them (and its own data), and then send packets to their successors. The sensor nodes are synchronized so that a sensor closer to the sink is triggered to aggregate and forward data at a later time. This distributed, iterative process continues until the sink receives and aggregates the final results.

The packet sent by v contains a 2-tuple (w_v, Θ) , where w_v is the weight of the aggregate in this packet, and Θ is the set of all parameters of the specific mixture model. The first field is computed recursively as follows:

$$w_v = \frac{\sum_{u \in PRED_v} w_u + 1}{(1-p) \cdot |SUCC_v|}$$

where p is the packet loss rate, $SUCC_v$ is the set of immediate successors of v in the routing graph, and $PRED_v =$

$\{u | v \in SUCC_u\}$ is the set of immediate predecessors of v . Notice that w_v approximates the weighted value for an aggregation value contained in the packet sent by node v . First, w_v is proportional to the sum of all weights from its predecessors plus one (counting v 's own data). Thus the weight reflects the effect of aggregation operation at v . Second, w_v is inversely proportional to the number of successors. Thus this weight also reflects the effect of splitting the results since in the next hop each successor will perform its own aggregation operation. Both are important since we need to make sure every individual value (by each node) should have an equal weight on the final outcome at the sink.

The packet loss rate p represents the approximate average loss rate. The calculation of w_v has also taken into consideration the effect of packet losses (and therefore prevents the loss of aggregation information). However, it is unrealistic to assume uniform packet loss rate in a sensor network, although uniform loss rate is just for the simplicity of our description. In practice, each node can monitor its own packet loss rate, or even the packet loss rate of every outgoing link. Then, we can replace the uniform loss rate in the expression with the estimated link loss rate.

Notably, at the sink s , the expression $\sum_{u \in PRED_s} w_u + 1$ represents approximately the total number of readings. In other words, we can approximate the duplicative-insensitive COUNT query by this value at the sink. Since we have taken the packet losses into consideration, the number provides an accurate and robust approximation.

3.4. Estimating Distribution Parameters

The key problem in our aggregation algorithm is how to estimate the distribution parameters. In this subsection we formulate the problem of estimating distribution parameters Θ , and describe the EM algorithm to solve the problem. We then present a case study: EM algorithm for Gaussian mixture model estimation.

3.4.1 The Problem of Parameter Estimation

Let us now focus on the parameters Θ . At each sensor node, the input information includes both the value acquired from its own sensing and the values received from its predecessors. The precisely-defined input distribution is

$$B_T(x) = \sum_{v' \in PRED_v} (\alpha_{v'} B_{v'}(x; \theta_{v'})) + \alpha_0 B_0(x; \theta_0)$$

where $\alpha_{v'} = w_{v'} / (\sum w_{v'} + 1)$ and $\alpha_0 = 1 / (\sum w_{v'} + 1)$ are the normalized weights. To generalize the value of the own reading of a sensor, we assume a specific single-value distribution B_0 with θ_0 .

From the view of an individual node, input data of a set of parameters leads to an output Θ_v , which is a set of parameters with smaller size. The problem is, a new distribution $B_v(x)$, derived from Θ_v , should constitute an approximation of the original distribution $B_T(x)$.

3.4.2 EM Algorithm for Parameter Estimation

To solve the problem of parameter estimation, we utilize the Expectation-Maximization (EM) algorithm, which is standard for finding maximum likelihood estimates of parameters in probabilistic models [11]. First, before the EM procedure, the measurements should be provided as the algorithm input x_1, x_2, \dots, x_k . For this purpose, we generate data from each input model. Because w_v approximates the weighted value for a data-set, for each $B_{v'}(x; \theta_{v'})$, it is easy to generate $w_{v'}$ independently and identically distributed (i.i.d.) random values with this distribution. Hence, in addition to its own reading, the node can obtain $k = \sum_{v'} (w_{v'})$ measurements. This random number generation method is efficient and accurate for the next EM algorithm.

In succession, let Z_{ij} be a random variable that equals one if and only if observation j comes from the i -th base distribution. We define a complete data log likelihood function:

$$Q(\Theta|\Theta^{(n)}) = \sum_j \sum_i Z_{ij} \left\{ \log \alpha_i^{(n)} B_i(x_j; \theta_i^{(n)}) \right\}$$

First we modify the complete data log likelihood function by replacing Z_{ij} with its conditional expectation, $E(Z_{ij}|x_i)$:

$$\sum_j \sum_i E(Z_{ij}|x_i) \left\{ \log \alpha_i^{(n)} B_i(x_j; \theta_i^{(n)}) \right\}$$

Notice that $E(Z_{ij}|x_j) = P(j \text{ from model } i|x_j)$. We may estimate $E(Z_{ij}|x_j)$ if we have preliminary estimates for $\Theta^{(n)}$. Then an individual re-estimation step that derives $\Theta^{(n+1)}$ from $\Theta^{(n)}$ takes the following form:

$$\Theta^{(n+1)} = \arg\max_{\Theta} Q(\Theta|\Theta^{(n)})$$

In other words, $\Theta^{(n+1)}$ is the value that maximizes (M) the expectation (E) of the complete data log likelihood with respect to the conditional distribution of the latent data under the previous parameter value.

The procedure of EM algorithm can converge fast by selecting the initial value $\Theta^{(0)}$ carefully. The weighted value for an aggregation value is contained in each node's packet. Hence, a better initial point can be provided even via a simple linear interpolation. A straightforward method is: $\theta_i^{(0)} = \sum_j (w_j \cdot \theta_{i,j}) / \sum_j w_j$. This initial point can reach the stable point fast after only a few iterations in our experiments.

3.4.3 The Special Case of GMM Estimation

The implementation of our algorithm focuses on Gaussian mixture model (GMM). We emphasize that the Gaussian distribution albeit representative is by no mean exclusive. With similar analysis, one can easily develop aggregation techniques based on other distributions, for example, Poison distribution and Erlang distribution. A standard Gaussian distribution function is as follow:

$$G(x; \mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp \left\{ -\frac{(x - \mu_i)^2}{2\sigma_i^2} \right\}$$

Consequently, for a κ -GMM, the model contains 3κ parameters. In succession, like usual EM algorithm, let Z_{ij} be a random variable that equals one if and only if observation j comes from the i -th Gauss. In a similar procedure, let t_{ij} denote an estimate for $E(Z_{ij}|x_i)$ under the preliminary estimates of α_i , μ_i , and σ_i :

$$t_{ij} = \frac{G(x_j; \mu_i, \sigma_i)}{\sum_k G(x_k; \mu_i, \sigma_i)}$$

We make a second modification to the complete data log likelihood function by replacing $E(Z_{ij}|x_j)$ with t_{ij} :

$$\sum_j \sum_i t_{ij} \left\{ \log \alpha_i - \frac{1}{2} \log(2\pi\sigma_i^2) - \frac{(x_j - \mu_i)^2}{2\sigma_i^2} \right\}$$

Treating the t_{ij} as fixed, we can calculate the values of α_i , μ_i , and σ_i^2 that optimize the twice modified complete data log likelihood function. The optimal values are as follows:

$$\begin{aligned} \alpha_i &= \sum_j t_{ij} / k \\ \mu_i &= \sum_j (t_{ij} x_j) / \sum_j t_{ij} \\ \sigma_i^2 &= \sum_j (t_{ij} (x_j - \mu_i)^2) / \sum_j t_{ij} \end{aligned}$$

These become our new estimates for α_i , μ_i , and σ_i^2 . Then we can alternate between re-estimating $E(Z_{ij}|x_j)$ (expectation step) and re-estimating α_i , μ_i , and σ_i^2 (maximization step) until the estimates become stable.

4. Analysis

The convergence behavior of the standard EM algorithm is examined in [9, 11]. We analyze our aggregation algorithm in terms of complexity and energy consumption.

4.1. Space and Time Complexity

Recall that each packet contains a 2-tuple (w_v, Θ) . If we use a double-precision floating-point number (8 bytes) to represent each field, the message size (our analysis are based on the assumption of GMM) would be $n_{size} = 8 + 24\kappa$ bytes, if we choose a κ -degree mixture model. This is still a large number. One method is to use floating-point numbers (4 bytes each) or even 2-byte integers to approximate the parameters. Certainly we can also pick a small value for κ . There is a tradeoff between space complexity (message size) and the accuracy of aggregation. That is, the more parameters we use, the more accurate results the system can provide.

Besides the space complexity, time complexity is also an important factor for an efficient aggregation algorithm. This is a drawback of the traditional EM algorithm. Although the

EM algorithm converges theoretically, its convergence speed is not stable. We reduce the time complexity and improve the convergence via the following approaches. First, an appropriate initial estimate is chosen as described in Section 3.4.2. This initial estimate represents a good knowledge of the foregone information. Thus, likely it leads to more accurate results. Second, we limit the maximum number of iterations to a constant. Thus, even in the worst-case scenario, each node runs the algorithm in constant time $O(1)$. Third, we reduce the computational complexity of each iteration of the EM algorithm. Recall the $E(Z_{ij}|x_i)$ estimation often requires exponential or logarithmic operations which are computationally expensive. Notice that these operations can often be approximated by a few addition and multiplication operations. For instance, to estimate t_{ij} in GMM, an exponential function can be written as an infinite series. This provides an opportunity to approximate the variable t_{ij} and to make each iteration less expensive.

4.2. Energy Consumption

We do not attempt to analytically derive the energy consumption of our algorithm for two reasons. First, our primary goal is to reduce the number and the size of packet transmissions as communication cost dominates the energy consumption. Therefore, the message (space) complexity has already indicated the level of energy consumption. Second, energy consumption also depends on the system configuration, which may differ for different sensor networks.

It is worth noting that there have been evidences to show that energy consumption due to computation (e.g., multiplication and other operations) is insignificant compared to the energy consumption due to communication. For example, in the RSA encryption experimented in [15], for the same size of data, the energy consumption due to computation is less than 1% of that due to transmission. Note each step in EM algorithm should contain fewer computations than in RSA encryption. Therefore, even though the EM algorithm needs a few more steps, its energy consumption is still negligible.

5. Experimental Evaluation

To evaluate the performance of our algorithm, we conduct series of experiments. We have implemented the simulator, and in this paper we quantify several performance aspects of our algorithm. We first describe our experimental evaluation methodology and then present the results and analysis.

5.1. Experimental Methodology

5.1.1 Network Topology

We use a network with 800 sensor nodes randomly distributed in a 1000×1000 square area. We consider both uniform distribution of the nodes and non-uniform distribution (also called skewed distribution). In particular, in the skewed

configuration, we assume 70% of all nodes are distributed in one half of the square area, and the other 30% in the other half. One of our experiments is to evaluate the performance under skewed node distribution. For the following experimental results, unless otherwise stated, the default setting is uniform node distribution.

The network topology is generated as follows. We assume all sensor nodes have a fixed radio range of 60 length units. If two sensor nodes are within the range of each other, they are considered neighbors. We assume this range does not change over time, and it is not affected by the interference during communication. We have found on average each node can communicate directly with about 6 neighbors at any point of time. This guarantees most nodes are in the largest connected components of the generated network. There are nodes disconnected from the rest of the network. Fortunately, they represent only less than 5% of all nodes and we ignore them. For some experiments, we need to vary the number of sensor nodes in the field. In such cases, we also vary the size of the square area in which the nodes are distributed to keep the node density almost constant.

Figure 1 shows the degree distributions for both the uniform node distribution configuration and skewed node distribution configuration. It can be seen that even with the uniform distribution, the nodes have varying degrees. Most of them have a degree between 5 and 11. With skewed node distribution, the node degree distribution is noticeably different. The curve with skewed node distribution is flatter than with uniform node distribution. Those nodes with high degrees are more likely to appear in one half of the square area, and meanwhile more nodes with low degrees will appear in the other half. In overall, with skewed node distribution, the node degrees are more variable.

We randomly choose a node as the sink in each experiment. The sink initiates a query by sending a query packet to all its neighbors, which forward this query to their neighbors further away from the sink using the GPSR [6] algorithm. In this way we construct a multi-path routing graph. Each node aggregates the results received from its predecessors and its own reading, and then sends the result to one or more of its successors along the routing path. It is clear that the larger the average degree, the more robust the routing algorithm is. For example, GPSR's greedy forwarding decision fails at some regions where the density of the nodes is low. We find that a system with an average degree of 6 is sufficiently large to support efficient data extraction for our algorithm.

5.1.2 The Data-sets

Our experiments are based on data-sets from a spatially-correlated elevation data released by USGS. We downloaded "susanville.gif" from <http://edc.usgs.gov/geodata/>. We select a 100×100 spatial subset of the original data as our data-set. This subset size can be easily scaled to cover the square where all nodes are distributed. Figure 2 shows the original

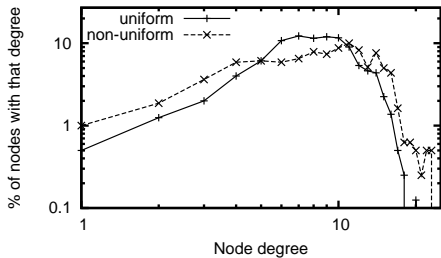
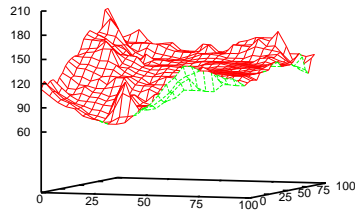
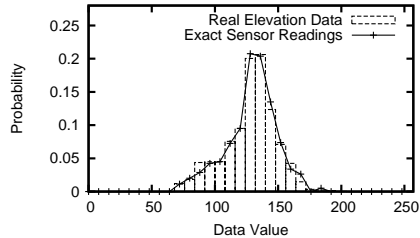


Figure 1. The node degree distributions of the networks.



(a) Elevation data-sets



(b) Data distribution

Figure 2. The original elevation data-sets used in our simulation and the distribution of the exact sensor readings.

elevation data and distribution of elevation data (exact sensor readings).

5.1.3 Algorithm Compared

We compare the performance of our algorithm (labeled as “GMM” in the figures) with that of q-digest [13] (labeled as e.g., “q-5%”). Q-digest is the only other work to provide approximate statistical information although only integers and finite data inputs are supported. In q-digest, each node aggregates the data it has received from other sensors into a fixed (user specified) size message. The message consists of a set of buckets of different sizes and their associated counts. Hence, each node has a separate q-digest structure which reflects the summary of data available to it. It uses a routing tree to forward this message.

To fairly compare different algorithms, first we carefully choose several parameter values. In q-digest, we set $k = 2$ and $\sigma = 256$, where k is the compression parameter (more details can be found in [13]) and σ is the bound value for input data and all the experimental sensor values must be less than this bound. Thus, the q-digest has a message size comparable to that of our 2-GMM implementation (unless otherwise stated, our implementation is based on 2-GMM model). Second, we round floating point numbers to integer values in q-digest. Finally, the default configuration includes a link failure rate of 5%. We also evaluate the performance of q-digest algorithm without packet losses for comparison (labeled as “q-0%” in the figures).

5.2. Experimental Results

In this subsection we compare the performance of our algorithm and the q-digest algorithm in the presence of link failures, node failures, as well as skewed node distributions. In the q-digest algorithm, we divide the data values into 32 equiwidth buckets and query all summaries to find the number of values in each bucket. For fair comparison, in our model-based approximation, we use the cumulative probability distribution to compute the value in each bucket (it is also a histogram-like form), *i.e.*, if a bucket range is x_0 and

x_1 , then the value at this bucket is $F(x_1) - F(x_0)$. To the best of our knowledge, no aggregation algorithms can provide approximation in continuous form so far. We believe this form is a suitable way to represent the data distribution and compare the performance of different algorithms.

5.2.1 Impact of Link Failures

Figure 3(a) shows the histogram results from our algorithm and the q-digest algorithm. In the figure, we plot the point at the begin of each interval. We can see that, even a relatively low link failure rate (5%) causes a large error in the results of the q-digest algorithm. Under the same condition, our algorithm provides a drastically more accurate approximation. We also observe that even with the 0% link failure rate, the q-digest approximation is not very good. That is because a traditional approximation technique like q-digest is based on the assumption that the data value is uniformly distributed. For instance, in the low level close to the root in the q-digest structure, the information obtained by a node is assumed to have come from all the leaf nodes uniformly. For that reason the curves of the q-digest algorithm’s results look *flat*, even though the input data does not follow the uniform distribution. We suspect this claim holds for any other traditional methods who assume uniform distribution of the data values.

We also compare the communication cost of the algorithms. Figure 3(b) shows the total communication cost. The communication cost is in the unit cost of transmitting one byte. While significantly improving the accuracy of approximation in the lossy environment, our multi-path GMM aggregation technique also introduces extra message overhead. It can be seen that the communication cost increases noticeably in our algorithm, although it increases only linearly with the number of sensor nodes. Overall, our duplicate-insensitive, fault-tolerant algorithm only results in a message overhead which is a constant factor times that of the q-digest algorithm. However, the communication cost is much lower than that of exact query techniques where a full list of sensor values must be sent back to the sink.

In addition we compare the error rates of the algorithms

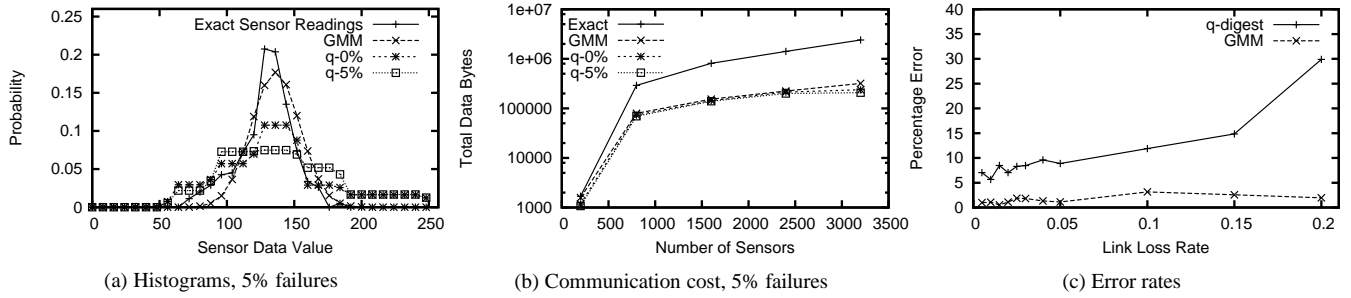


Figure 3. Performance comparison of the algorithms in the presence of link failures.

	With link failures		With node failures		With skewed distribution	
	CE	ME	CE	ME	CE	ME
q-5%	26.12%	5.05%	33.63%	11.81%	19.75%	7.36%
GMM	2.85%	1.11%	4.00%	4.23%	4.89%	2.67%

Table 1. Error rates of the algorithms when COUNT and AVERAGE aggregates are computed. “CE” denotes error rates on count and “ME” denotes error rates on mean.

when the link failure rate increases. Figure 3(c) shows the effect of link failure rate on the performance (accuracy) of each algorithm under study. The error is defined as the ratio between the difference (between a sample and the true value) and the true value. That is, for a sample value x and correct value \bar{x} , the error is defined as $(|x - \bar{x}|)/\bar{x}$. The figure shows that, when computing the AVG aggregate, the error rate of q-digest grows quickly as the link failure rate increases. On the other hand, the error rate of our algorithm is still low since our algorithm takes the link loss rate into consideration. Another observation is, the q-digest algorithm has relatively high error rate even when the link failure rate is very low.

Table 1 shows the relative error rates of the algorithms when there are link failures and the COUNT and AVG aggregates are computed. See the two columns on the left of the table. The q-digest incurs high error rate (about 26%) for the COUNT aggregate in lossy environments. Our algorithm exploits multi-path routing (and aggregation), and it is robust (error rate = 2.85%). For the AVG aggregate, our algorithm leads to very low error rate too compared to q-digest.

5.2.2 Impact of Node Failures

Node failures are potentially more severe than link failures since a single node failure usually means multiple link failures. We would like to know the impact of node failures on the performance of the algorithms. Hence, we compare them when node failures occur, but the other parameters are the same as in the previous experiments. Figure 4 shows the results when the node failure rate is 5% (in this figure q-5% represents a node failure rate of 5% instead of link failure rate). For the q-digest algorithm we consider only the case with 5% node failures. Clearly, this figure shows that node

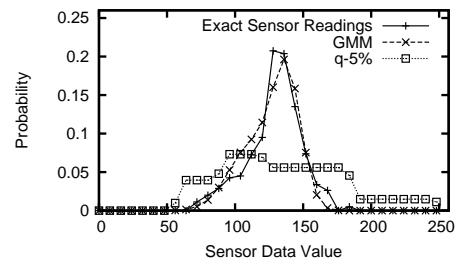


Figure 4. Exact and approximate histograms in the presence of 5% node failures.

failures hurt the performance of the q-digest algorithm drastically. On the contrary our algorithm is more robust. Table 1 gives their error rates in the two middle columns.

5.2.3 Impact of Skewed Node Distribution

We distribute the sensors in a skewed fashion. This skewed distribution leads to more variable node connectivity and more clustered network. We then study whether such a skewed node distribution would affect the performance of the algorithms. Figure 5 shows the results. There is no significant difference compared to the results from the previous experiments. A slight difference shown in Table 1 is, our algorithm obtains counts that depart a little more from the true values. We suspect it is because of the higher variability of the node degree. Since there are more nodes with low degrees, packets can be lost along their links. Therefore, it is possible our algorithm does not fully recover from the losses. In overall our algorithm does reasonably well.

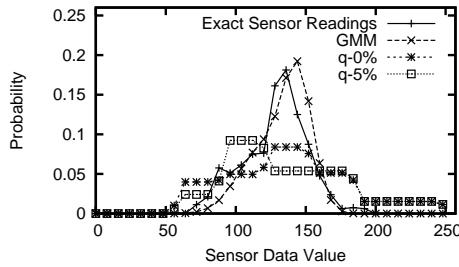


Figure 5. Exact and approximate histograms in the presence of skewed node distribution and 5% link failures.

6. Conclusion

We have presented a novel algorithm for in-network data aggregation in sensor networks. The algorithm is scalable: it propagates statistical information rather than individual values in the network, and hence it reduces the network communication cost even in large-scale networks. The algorithm is robust: it exploits loss-tolerant multi-path approaches to data aggregation. Thus it extracts the statistical information of the original data distribution, but preserves the accuracy of estimation and prevents the loss of valuable statistical information. It is the first study on using mixture models to approximate sensor data distributions. We have demonstrated that this is a viable approach to providing more accurate results. It outperforms a previous approximation algorithm by a large margin in most configurations.

There are several directions in the future. First, we are seeking the possibility of using more efficient algorithm to reduce the computational overhead of our aggregation technique. In particular, the original EM algorithm can be expensive for those sensor nodes with low computation capacity. We would like to exploit approximation algorithms who can perform as well but less expensively. Second, we would like to exploit more multi-path routing protocols for our aggregation algorithm. Currently, geographic routing protocols is used but other routing protocols can also be used. We hope to evaluate and improve our aggregation algorithm when other protocols are adopted.

Acknowledgments The authors would like to thank Meral Ozsoyoglu for her feedbacks on query processing techniques in sensor networks.

References

[1] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)*, April 2004.

[2] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing historical information in sensor networks. In *Pro-*

ceedings of ACM SIGMOD International Conference on Management of Data, June 2004.

[3] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proceedings of Conference on Very Large Data Bases (VLDB)*, August 2004.

[4] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM Mobile Computing and Communications Reviews*, 5(4), 2001.

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom)*, August 2000.

[6] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom)*, August 2000.

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

[8] S. Nath, P. B. Gibbons, Z. R. Anederson, and S. Seshan. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2004.

[9] R. Nowak. Distributed EM algorithms for density estimation and data clustering in sensor networks. Technical Report TREE0203, Department of Electrical and Computer Engineering, Rice University, 2002.

[10] S.-J. Park, R. Vedantham, R. Sivakumar, and I. F. Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2004.

[11] Y. Pawitan. *In All Likelihood: Statistical Modelling and inference Using Likelihood*. Clarendon Press, 2001.

[12] N. Sadagopan and B. Krishnamachari. Maximizing data extraction in energy-limited sensor networks. In *Proceedings of IEEE INFOCOM*, March 2004.

[13] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2004.

[14] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman. Wavescheduling: energy-efficient data dissemination for sensor networks. In *Proceedings of International Workshop on Data Management for Sensor Networks (DMSN)*, August 2004.

[15] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Proceedings of IEEE International Conference on Pervasive Computing and Communication (PerCom)*, March 2005.

[16] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, June 2002.

[17] Y. Yao and J. Gehrke. Query processing for sensor networks. In *Proceedings of Conference on Innovative Data Systems Research (CIDR)*, January 2003.