Febuary 11, 2004 TR CS-04-04

Accuracy Vs Lifetime: Linear Sketches for Appoximate Aggregate Range Queries in Sensor Networks

Konstantinos Kalpakis, Vasundhara Puttagunta and Parag Namjoshi

Computer Science and Electrical Engineering Department
University of Maryland Baltimore County
1000 Hilltop Circle
Baltimore, Maryland 21250

[1]Email: {kalpakis,vputta1,nam1}@csee.umbc.edu
[2]Supported in part by NASA under Cooperative Agreement NCC5–315.

Accuracy Vs Lifetime: Linear Sketches for Appoximate Aggregate Range Queries in Sensor Networks[1,2]

Konstantinos Kalpakis, Vasundhara Puttagunta and Parag Namjoshi

Abstract

Query processing in sensor networks is critical for several sensor based monitoring applications and poses several challenging research problems. The in–network aggregation paradigm in sensor networks provides a versatile approach for evaluating simple aggregate queries, in which an aggregation–tree is imposed on the sensor network that is rooted at the base–station and the data gets aggregated as it gets forwarded up the tree. In this paper we consider an two kinds of aggregate queries: *value range queries* that compute the number of sensors that report values in the given range, and *location range queries* that compute the sum of values reported by sensors in a given location range. Such queries can be answered by using the in–network aggregation approach where only sensors that fall within the range contribute to the aggregate being maintained. However it requires a separate aggregate to be computed and communicated for each query and hence does not scale well with the number of queries.

Many applications in sensor networks require just approximate query results. In this paper, we consider an alternate approach inspired by techniques developed for OnLine Analytical Processing (OLAP) and stream computations, to get longer lifetimes for a small loss in accuracy. The idea is to maintain a summary data–structure over the sensor data using in–network aggregation, and use it to answer queries at the base station in an approximate manner. We construct sketch based summary data–structures called *linear sketches* that are optimized for answering aggregate range queries effectively and accurately. While accuracy of the query results increases with the size of the sketch, the lifetime of the sensor network decreases, therefore there is a trade–off between accuracy of the query results and lifetime of the sensor network. We performed several experiments to evaluate the proposed method against other commonly used methods. Our results show that linear sketching achieves significant improvements in lifetime of sensor networks for only a small (acceptable) loss in accuracy of the queries. Further, the accuracy of the queries is observed to be significantly better than results using other classical techniques using Discrete Fourier Transform and Discrete Wavelet Transform.

1 Introduction

There is an ever–growing demand for continuous monitoring of varied physical environments. Applications include, habitat and environment monitoring, military surveillance, terrain exploration, biomedical applications, traffic monitoring and building environment control systems to mention a few. Rapid advances in micro–sensor (MEMS) technology and low power digital/analog electronics have led to the development of distributed, wireless networks of sensor devices for such activities. Sensor networks consist of hundreds of low cost nodes that come with wireless communication and certain processing and storage capabilities in addition to sensing capabilities. These nodes are usually powered by a non–replenishable battery and are therefore severely energy–constrained. The nodes are deployed in the *sensor field* and can be thought of as distributed streaming data sources. Therefore, sensors provide a distributed view of the field being monitored. The goal of the sensor network is to collect information from the sensors, that is required for supporting various (monitoring) applications at a resource rich base–station. In doing so, the sensor network should work in an energy efficient manner, i.e. in a manner that will enable sustaining itself for the longest period of time.

To this end, several energy aware data–gathering (or collecting) techniques have been developed for sensor networks. Here the idea is to systematically gather the sensed data from each sensor, so that it is eventually transmitted to the base–station for further processing. Chang and Tassiulas [4, 5] give a flow–based approach to gather data in an optimal manner. However this is very expensive in terms of the energy consumed by the sensor network because of the size of the raw data that must be communicated. Consequently the sensor network may be functional only for a short duration. Fortunately, it is often not the individual raw data from the sensors that matters to most applications, but certain information in the form of summaries or aggregations of the data that is of more relevance [19]. Data fusion or aggregation allows for in–network aggregation of data, i.e. combining of packets from different sensors while enroute to the base–station. Therefore rather than the traditional approach of gathering all the raw data from the sensors at the base–station and then computing the aggregate, in–network aggregation utilizes the processing power of the sensors for aggregating the sensor data as they get collected during routing. This eliminates redundant (raw–data) transmissions and thereby helps conserve energy and at the same time provides the base–station with vital statistics of the environment that is being monitored. Communication cost is the dominant factor in determining the lifetime of the sensor network. In–network aggregation takes advantage of processing capability of the sensors to save on the communication costs. In–network aggregation of data has been shown to dramatically increase the lifetimes of sensor networks (see for e.g.Krishnamachari et al. [16]). For this reason, data fusion or aggregation has emerged into a core service supported in sensor networks (for e.g. TAG by Madden et al. [21]). Several studies have been carried out toward performing in–network aggregation in sensor networks [10, 12, 14, 15, 17, 18]. The main focus here is to come up with good data aggregation trees along which in–network aggregation can be carried out. In Pegasis [17], sensors form chains so that each sensor transmits and receives from nearby sensors. The aggregate is computed along the chain and eventually reaches the base station. Nodes take turns to transmit to reduce the average energy spent by any single sensor, which leads to different chains in different rounds. Lindsey et al. [18] propose a hierarchical scheme based on Pegasis for data–aggregation. Kalpakis et al. [14] propose the Maximum

Lifetime Data Aggregation [14] algorithm for data aggregation that gives a near optimal data–aggregation schedule consisting of aggregation trees. Further, Kalpakis et al. [15] present a computationally efficient approximate scheme for maximizing lifetime by intelligently selecting aggregation trees from a given candidate set of trees. For the candidate set of trees they use the randomly mutated trees produced from the LRS trees from [18].

1.1 Query Processing in Sensor Networks

From databases perspective, in–network aggregation is a mode for evaluating aggregate queries over the sensor network. Sensor network applications rely heavily on aggregate queries. Queries are also considered a natural way for users to interact with the sensor network [23]. Therefore, several studies have been carried toward (aggregate) query processing in sensor networks [20, 21, 23]. The common approach to query processing in sensor networks has two phases. Users specify queries at the base–station using an SQL–like query language. During the *dissemination phase* the query layer interacts with the network layer for dissemination of the queries to the sensors. Then, during the *aggregation phase*, the sensor data is gathered and aggregates are computed for each query as it is gathered using in–network aggregation along the aggregation–tree. In the case of continuous queries, where we want answers to queries periodically, the aggregation is performed periodically and each time period is referred to as a *round*.

For example consider the following query for finding the number of sensors in the sensor field that record temperature between 10^oF and 30^oF.

select count(*)
from sensors
where $10 \leq$ temperature ≤ 30.

First, this query is injected into the sensor network to all the sensors. Then aggregation is performed along the aggregation tree as follows. Every node waits for the counts from each of its child–nodes. It adds them all up contributes its own count toward the COUNT only if it satisfies the where–clause and forwarders this to its parent–node. This approach allows us to answer queries accurately. However, it has several drawbacks. (a) Every query needs to be disseminated to the sensors and this is an expensive operation because of the communication involved. (b) Every single query is evaluated separately. This means that multiple aggregates need to be maintained, one for each query. This severely restricts the number of queries that can be answer before the sensor network dies. Also, certain sensors may be drained off energy much quicker than other sensors because of frequent querying of particular sensors.

To overcome these drawbacks, we propose an alternate approach that is inspired by OnLine Analytical Processing (OLAP) and is the approach taken in the AQUA project [7] for query processing over data streams. The idea is to maintain and update a small space summary data structure as the data appear in the stream and use the summary data structure to directly answer queries quickly in an approximate manner. The constraints that are imposed by streams (see Babu and Widom [3]) make techniques for stream computations suitable for sensors. These summary data–structures are typically computed as decomposable aggregates, i.e. they can be expressed as an aggregation function f over sets a and b so that $f(a \cup b) = g(f(a), f(b))$, where g is called the combine–function that is computable in small space and time. Decomposable functions are suitable for the in–network aggregation paradigm, therefore a similar approach can be taken for answering queries in sensor networks. During each round, the summary data structure can be computed over the data from all the sensors using in–network aggregation. The user queries at the base–station can then be answered directly from this summary data–structure at the end of each round. This way, the lifetime of the sensor network is not limited by the number of queries. However, it is limited by (a) the amount of computation at each sensor to compute the synopses which determines the number of CPU cycles and the corresponding energy, (b) the space complexity for computing the synopses which determines the energy required to power memory on the sensor, and most importantly (c) the size of the synopses which determines the energy for transmitting and receiving using wireless radio. Usually there is a direct relationship between the size of the synopses and the accuracy of the queries and since most monitoring applications are interested in approximate query answers, **this approach offers a viable option for increasing the lifetime of the sensor network for a small loss in accuracy of the query results**.

Several such data structures have been proposed for query answering over streaming data. As mentioned earlier, these techniques fit well into the in–network aggregation paradigm for sensor networks. It is important to note that the synopses data structure to be computed depends on the application at hand and in our case, the kind of queries we wish to answer. Several data structures (see for e.g. Gibbons and Matias [8]) have been proposed in literature. For example, small space samples and histograms are popularly used for answering selectivity and aggregate range queries and to compute the size of joins and number of distinct elements. Recently, there have been several sketch based summary data structures. Thaper et al. [22] present sketches based on randomized projections that are used for answering range queries. Guha et al. [9] compute wavelets over streams for constructing histograms.

In the sensors domain, special purpose aggregates have been proposed. For example, Cosidine et. al. [6] present approximate sketches for COUNT, SUM and AVG that are robust with respect to node failures,etc. However, these aggregates are over all the sensors. Hellerstein et al. [11] argue that monitoring applications demand more sophisticated aggregate query processing over sensor networks. They compute wavelets over the sensor data that can be used to answer approximate aggregate range queries.

We consider the following two important kinds of aggregate range queries.

1. *Value Range Queries*: How many sensors recorded values in a particular range?

```
select count(*)
from sensors
where 10 ≤ reading ≤ 20.
```

2. *Location Range Queries*: What is the aggregate of values recorded by sensors located in a given rectangular region(range)?

   select average(reading)
   from sensors
   where $10 \leq x \leq 20$ and $10 \leq y \leq 20$.

Aggregate range queries occur commonly in sensor based monitoring applications and form an important class of aggregate queries. In several applications including monitoring applications over sensor networks, exact query answers are not very important and approximate query answers are sufficient. Therefore we consider approximate aggregate range queries in this paper. Histograms are the popularly used summary data structures for answering such queries and several sketch based approaches for histograms have been suggested in literature [9, 22]. However these methods are designed to capture the norm of the histogram. They depend solely on the data and do not consider the queries at all. Often there is some information about the queries. For example certain regions (ranges) could be more interesting and hence queried more often. Further, in the case of monitoring applications, it is common to have continuous queries, in which case, the same range query is posed in every round. Therefore it is of great interest to design sketches for a specific set of aggregate range queries. In this paper we propose to use linear sketches [2] that we developed for answering approximate aggregate range queries effectively and efficiently over sensor networks to get improved lifetimes for a small loss in accuracy of the results.

1.2 Road Map

In section 2 we describe the linear sketching approach to answering range queries and describe how value and location range queries can be answered in sensor networks. We evaluate our approaches experimentally and present results in Section 3 and conclude in section 4.

2 Linear Sketching Approach

2.1 Linear Sketching Preliminaries

In this section we present definitions, notation and some results on linear sketches. For detailed discussion and proofs please refer to [2]. We begin with a few definitions.

Definition 1 *Let A be a matrix over $\mathcal{C}$. The complex conjugate of A is denoted by $\overline{A}$. The conjugate transpose A^* of A is defined as $A^* = \overline{A}^T$.* ■

Definition 2 (STANDARD INNER PRODUCT)
For any two complex vectors $x, y \in \mathcal{C}^n$, the standard inner product is the scalar given by $\langle x, y \rangle = y^* x$. ■

Definition 3 (LINEAR SKETCH)
Let P be an $N \times k$ complex projection matrix, with columns $p_1, p_2, \cdots, p_k$ so that $P = [p_1 p_2 \cdots p_k]$. We call P a sketching matrix. Let x be a vector in $\mathcal{C}^N$. The projection of x onto p_i is given by $\langle x, p_i \rangle$, $i = 1, 2, \ldots, k$. The (linear) sketch or projection of x with respect to the sketching matrix P is the k–dimensional vector given by $\widehat{x} = [\langle x, p_1 \rangle \langle x, p_2 \rangle \cdots \langle x, p_k \rangle]^T$. Equivalently,

$$\widehat{x} = P^* x \tag{1}$$

The sketch of an $N \times m$ matrix A with respect to the sketching matrix P is a matrix whose columns are equal to the sketch of the corresponding column of A, i.e.

$$\widehat{A} = P^* A. \tag{2}$$

■

Let x be a $N \times 1$ data vector. Consider a query vector q of the same length that is evaluated against x using the standard inner product as follows.

$$a = \langle x, q \rangle = q^* x \tag{3}$$

For example, when q is a 0–1 vector, Equation 3 simply sums the components of x where there are 1's in q. We refer to such queries as *sum–queries*. Let Q be an $N \times m$ matrix whose columns correspond to m such sum–queries. Then from Equations 1 and 3. we can see that $\widehat{x} = Q^* x$ yields a vector of length m, whose components are exact answers to the m queries in the query matrix Q. In other words, when we use the query matrix Q as the sketching matrix, the sketch $\widehat{x}$ of the data vector gives the exact answers to all the queries. In general, the number of queries m can be very large making the sketch too big. Cost of computing the entire sketch as well as communicating it increases with the number of queries. For this reason we explore the following idea: *Can we find just a small number (k) of special queries, whose answers we can maintain accurately, so that we can use their answers to answer all the queries in Q in an approximate manner?*

Suppose that these k special queries form our sketching matrix P. Then the sketch of the data vector will contain exact answers to the special queries. We need a way to compute answers to the actual queries in Q. An approximate answer to the query q with respect to vector x can be estimated using their sketches $\widehat{q}$ and $\widehat{x}$ respectively as follows.

$$\widetilde{a} = \langle \widehat{x}, \widehat{q} \rangle = \widehat{q}^* \widehat{x} \tag{4}$$

From signal processing perspective, when the sketching matrix P contains the Fourier vectors, then $\widehat{x}$ contains simply the Fourier coefficients of x. Similarly when P contains the wavelet vectors, $\widehat{x}$ gives the respective wavelet coefficients of x. The standard practice is to maintain the top–k coefficients of the data vector which implicitly means having vectors corresponding to the top–k coefficients as the sketching matrix. These approaches rely on the ability to capture the norm of the data vector using the top–k coefficients and do not make use of the queries at all. (The projection matrix depends only on the data vector.) Further, in the case of streaming data, the errors in estimating query results escalate with the error in estimating the top–k coefficients.

However, queries often have certain patterns and structures that can be exploited to get better results. For this reason, we propose to use an alternate approach of using sketching matrices that depend on the queries. We use the sketching matrix whose vectors are the top–k eigenvectors of the matrix QQ^*. Such a sketching matrix minimizes the mean squared error with respect to the queries at hand. Please refer to [2] for the error bounds. Further, in the case of range queries, the query vectors contain blocks of consecutive ones. If Q contains all range queries of a fixed extent (i.e. query vectors with fixed number of consecutive ones) then the eigenvectors of QQ^* are indeed the Fourier vectors. Therefore we can have the sketching matrix P to be the certain Fourier vectors corresponding to the top–k eigenvalues of QQ^* and then the sketch $\widehat{x}$ is simply the corresponding Fourier coefficients of x. Note that the choice of the Fourier coefficients depends solely on the query matrix Q and may not be the same as the top–k Fourier coefficients of the data vector x. The advantage of Fourier vectors is that they have a succinct representation and do not have to be explicitly stored as any element of the sketching matrix can be generated on the fly whenever required using small space and time. Note that even if the query that needs to be evaluated is different from those in the query matrix Q, we can still evaluate it approximately from the sketches with reasonable errors. (see [2] for detailed experiments). In the remaining part of the section, we show how the above results can be applied to answer value range queries and location range queries over sensor networks.

2.2 Answering Value Range Queries

Let n be the number of sensors. The frequency distribution of the data collected from all the sensors in one round, gives the number of sensors recording a particular value in that round. If we can maintain the frequency distribution of values from all the sensors we can answer value range queries accurately by simply adding the frequencies reported at the values in the query range.

We assume that the values recorded by sensors take integer values between 1 and M. Therefore, frequency distribution can be thought of as an $M \times 1$ column vector, h. Similarly, a value range query can then be thought of as a 0–1 column vector q of the same size, with 1's in the range of the query and 0's everywhere else. Then the exact answer to the range query can be answered using Equation 3 as $a = \langle h, q \rangle = q^* h$. To use this approach we need to be able to compute the frequency distribution, h over the sensor network. We observe that, if we consider each of the sensors generating an $M \times 1$ column vector with a 1 at the index corresponding to the value it measures, then h is simply the sum of such vectors from all the sensors. Let

$h^{(i)}$ be the vector generated by the i^{th} sensor, then

$$h = \sum_{i=1}^{n} h^{(i)} \tag{5}$$

In other words, h is a simple *SUM* aggregate of the vectors generated at each sensor and can therefore be computed using any in–network data–aggregation scheme in sensor–networks. However, this approach will need packets of size $O(M)$ to be communicated and aggregated which may be infeasible due to energy constraints.

We propose to apply the linear sketching approach mentioned previously for this problem. Because sketching is a linear function, the sketch $\widehat{h}$ using the sketching matrix P is simply the sum of sketches of the individual vectors generated at each sensor. i.e.

$$\widehat{h} = P^* h = P^* \sum_{i=1}^{n} h^{(i)} = \sum_{i=1}^{n} P^* h^{(i)} = \sum_{i=1}^{n} \widehat{h}^{(i)} \tag{6}$$

Therefore, just as h is a sum–aggregate over the vectors generated at each sensor, $\widehat{h}$ is also a sum–aggregate over the sketches of these vectors using the sketching matrix P and can therefore be carried out over the sensor–network using any data–aggregation service.

Also note that for computing the individual sketch at each sensor, we need to have the matrix P at each sensor. Further, since there is only a single non–zero component in any $h^{(i)}$, computing the sketch requires only $O(k)$ multiplications. In the case of all range queries of fixed size, the projection matrix P is simply certain Fourier vectors, and therefore does not have to be explicitly stored. If we consider a b–byte representation for each component of the sketch we will need to communicate and aggregate packets of size $(bk + p)$ bytes where p is the header size. Typically $k << N$. We consider $p = 8$ and $b = 2$. The actual queries are evaluated at the base–station using the sketch $\widehat{h}$ and the sketch of the query vector as shown in Equation 4.

2.3 Answering Location Range Queries

Location range queries are the queries that ask for the sum of values reported by sensors located in a particular (spatial) range. In this case, we consider data generated by all the sensors as a single vector that is indexed by sensor–id. Therefore the size of the vector is the same as the number of sensors n. When we consider any ordering of the sensors, the range query translates into a sensor-query, i.e. a 0–1 vector with 1s in the place where the corresponding sensor is located within the query range. The sensor–queries need not contain blocks of consecutive 1's. Therefore, the sketching matrix in this case is not the Fourier matrix and may not have a succinct representation. We use the sketching matrix that corresponds to the top–k eigenvectors of QQ^*, where Q is a a query matrix obtained sensor–queries corresponding to the location range queries.

3 Experimental Evaluation

We perform experiments to observe the trade-off between the gain in lifetime of a sensor network and loss in accuracy in estimating aggregate queries over sensor networks using in–network aggregation. The results presented in this section are based on the following experimental setup.

3.1 Experimental Setup

We have a synthetically generated field in which sensors are randomly placed and there is a single base–station. Queries are posed at the base–station. The readings of the sensors vary over time. In every round, data from the sensors is gathered or aggregated (depending on the approach) and the queries are evaluated at the base–station. We estimate the lifetimes of the sensor networks. We assume that the energy for communication dominates the energy used for processing and ignore the computation cost.

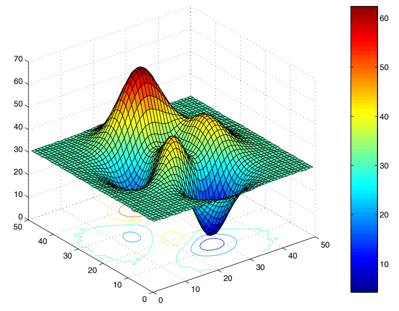
In the next part of the section we present a detailed description of the experiments.

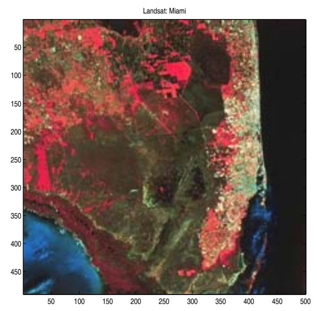
3.2 Sensor Fields

Field Sizes and Sensor Placements: We consider fields of two sizes *Field 1* is a $50m \times 50m$ with 50 sensors and base station at (25m,150m) and *Field 2* is a $100m \times 100m$ with 500 sensors and base station at (50m,300m). The sensors are randomly placed in the field according to a uniform random distribution.

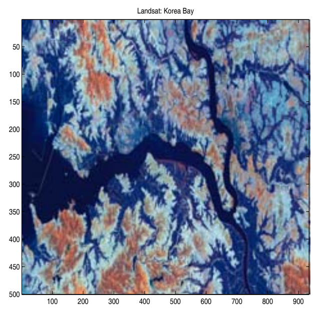
Field Value Distribution: The field that is to be monitored by the sensor network is expected to be spatially correlated. Further, very often, the changes in the environment are periodic in nature. To account for the spatial correlation, we consider two intuitive approaches for the initial field distribution. The first approach is to have the field according to a mixture of Guassians. This way we can have the spatial correlations in the simulated sensor field. Also, pictures (or photographs) have a strong spatial correlation among their pixel values. So, our second approach is to have a picture with the pixel values corresponding to the initial field values. The initial fields that we consider are the following.

1. **Synthetic50** This is generated using four Gaussians and is illustrated in figure 1(a). **Synthetic50** is the initial field value distribution for Field 1.
2. **LANDSAT Miami** : This data set is Landsat Thematic Mapper image of Miami, Florida [1] that is in figure 1(b). We consider a 100×100 pixel piece of the second band of this image for initial Field 2 values.
3. **LANDSAT Korea** : This is a Multi-spectral (Landsat) image of the Korean Bay with Bands 5,4,3 (RGB) [1] that is in figure 1(c). We consider a 100×100 pixel piece of the second band of this image for initial Field 2 values.


(a) **Synthetic50** : $50m \times 50m$ field



(b) Full image of LANDSAT Miami



(c) Full image of LANDSAT Korea Bay

Figure 1: Intial fields

The initial field value distributions are used to generate different values over time so that they exhibit some periodicity as well as randomness as follows. The value sensed by a sensor is the value of the field at the cell in which it is located. The value sensed by the i^th sensor at time t is given by $X_i(t)$. We represent the values at time $t = 0$, $X_i(0)$ as μ_i. These come from the initial fields. Then the values at each sensor change with time as a sine curve according to the equation 7.

$$X_i(t) = \mu_i + a_i \sin(\frac{2\pi t}{\tau_i} + \phi_i) + \varepsilon_i \tag{7}$$

Here a_i, τ_i and ϕ_i are amplitude, frequency and phase for the change in values sensed by sensor i. These are time invariant for all the sensors and are chosen as follows. $a_i = \mu_i/3$. $\tau_i = 60$. ϕ_i is chosen randomly according to a uniform distribution in $(0, \pi)$. ε_i is a random variable that corresponds to white noise with variance $(\mu_i/10)^2$.

3.3 Queries

The queries are posed at the base–station. All queries that arrived during the current round are evaluated at the end of the current round. In the case of continuous queries, the same query is posed in every round as long as the query is active. We consider two kinds of aggregate range queries for our experiments.

Value Range Queries : Value Range Queries are queries of the 1st type. They specify a range over values and ask for the number of sensors that record values in that range. We consider value queries of *fixed extent* ($ValQFixed(v)$), which is a set all queries of fixed extent v over the value domain. We also consider value queries of*random extent* ($ValQRand(\mu, \sigma^2)$) which is a set of 100 queries whose extent is chosen randomly from a normal distribution with a mean μ and variance σ^2.

Location Range Queries : Location Range Queries are queries of the 2nd type. They ask for sum of values recorded by sensors that are located in a particular rectangle in the field. We consider location queries of fixed extent ($LocQFixed(v)$) which is the set of all queries of fixed $v \times v$ spatial extent. We consider only queries that contain at least 1 sensor.

3.4 Approaches

We compare both data–gathering based approaches (without aggregation) as well as data–aggregation based approaches for evaluating the lifetimes and accuracy of queries in our experiments. We compare results from the following approaches for evaluating queries.

No Aggregation (NoA): This is a data–gathering approach in which each sensor sends its value to the base station at the end of each round. The values recorded by each sensor are simply forwarded without any data aggregation. We use the flow–based algorithm by Chang and Tassiulas [4] for routing. We assume that the data is 2 bytes and the header is 8 bytes. Therefore the packet size is 10 bytes. A complete histogram is computed at the base–station from the raw data collected from each of the sensors and used to answer the queries. This approach obviously gives exact answers to all the queries and the norm will also be exact for each round. However the number of rounds that the network lasts using this approach is expected to be much smaller than other approaches.

Next we describe the data–aggregation based approaches we use for estimating queries. For evaluating the lifetimes using these approaches we use data–aggregation techniques by Kalpakis et al [13, 15]. For experiments with small number of sensors (Field 1) we use the MLDA [13] approach that gives near optimal lifetimes for data aggregation. For larger sensor networks (Field 2), we use the A–MLDA approach [15]. The packet size for these methods depends on the approach for evaluating queries that are described next.

Naive Aggregation (NA): Sensor data is aggregated in a naive way. The packet has n slots where n is the number of sensors. There is a slot for each sensor in the packet that is filled by the corresponding sensor. Data aggregation can be performed in the obvious way as an addition of two $n \times 1$ vectors. This approach maintains the exact histogram, therefore the queries will be answered exactly, however the packet sizes are too large due to which the lifetime of the sensor network suffers severely. We assume two bytes for each sensor value and 8 bytes for the header. Therefore, the packet size in this case is $2n + 8$ bytes.

Haar Wavelets (DWT): In this technique, the top–k Haar wavelet coefficients are maintained using data–aggregation. Normally one does not know which coefficients will be the top–k and hence we do not know before hand which coefficients to maintain. Therefore one has to estimate the top–k Haar coefficients using in–network aggregation (for e.g. by using approaches suggested by Hellerstein et al. [11] or Guha et al. [9]). In the experimental results we present only results using the true top–k DWT coefficients. (The results with the estimated top–k DWT coefficients are much worse and are not reported for brevity.) With header size of 8 bytes and 2 byte representation for each coefficient, the packet size in this case is $2k + 8$ bytes, where k is the size of the sketch.

Discrete Fourier Transform (DFT): This method is similar to DWT, except that we use the top–k DFT coefficients of the data vector.

Linear Sketching (LS): In this method we use the linear sketching approach to compute the sketch of of size k. In the case of fixed extent queries we use the sketching matrix due to all queries of the same fixed size. In the case of random queries, we use the sketching matrix due to all queries of a fixed extent that is the same as the mean extent used to generate the queries. We use 2 byte representation for each component of the sketch. Therefore the size of the packet in this case will be $2k + 8$ bytes (8 byte header).

3.5 Quantitative Performance Measures

We use the following quantities to measure the performance of different techniques.

MLT : Mean LifeTime of the sensor network as the number of rounds it lasts until the first sensor drains out of energy, where the mean is computed over 20 different placements of sensors in the field.

MSE : Mean of Squared Errors of results to all the queries over all the rounds and over all the placements.

RLE : Mean Relative Errors of results to all the queries over all the rounds and over all the placements.

REN : Relative Error in the norm of the frequency distribution over all the rounds and over all the placements.

3.6 Experiments

Experiment 1: We consider a field with 20 different sensor placements and report the MLT over the different placements. For the first set of experiments, we use the query set $ValQFixed(v)$ for 60 rounds over each placement with v as 10% of the range that the values that the sensors record and report MSE, RLE and REN. The sizes of the domains for **Synthetic50** , **LANDSAT Miami** and **LANDSAT Korea** are 101, 354 and 387 respectively, therefore we use $ValQFixed(10)$, $ValQFixed(35)$ and $ValQFixed(39)$ for the three datasets respectively. For the NoA method we use the approach by Chang and Tassiulas [4, 5] to compute the lifetimes. This approach gives exact results, therefore, the MSE, REL and REN are always zero for this approach. For the NA and NoA approaches, the errors are always 0. For the remaining aggregate based approaches we present the results obtained for different sketch sizes. The results for the three datasets are presented in tables 1, 2 and 3 respectively. We note that as the sketch size increases, the errors in estimating the range queries using different aggregation techniques decreases. The LS method gives significantly lesser errors in estimating answers than the classical methods of DFT and DWT. Note that the errors for DFT and

Table 1: Results for experiments with $ValQFixed(10)$ over the dataset **Synthetic50** . The number of sensors $n = 50$; the field size is $50m \times 50m$; the base station is located at (25m, 150m). The fixed extent v is 10% of the size of the domain. The exact answer averaged over all queries for 60 rounds and 20 placements is 15.15. For NoA, packet size in bytes, $p = 10$ and for NA, $p = 2n + 8$. Both NoA and NA approaches always give exact results. For DWT, DFT and LS k is the size of the sketch and the size of the packets, $p = (2k + 8)$ bytes.

Method	p	LifeTime	MSE			RLE			REN		
NoA	10	20,716	0			0			0		
NA	108	10,438	0			0			0		
Other Aggregation methods (DWT/DFT/LS)											
k	p	LifeTime	MSE			RLE %			REN %		
			DWT	DFT	LS	DWT	DFT	LS	DWT	DFT	LS
1	10	112,742	13.86	3.73	3.73	94.15	25.25	25.25	89.40	31.20	31.20
3	14	80,530	11.75	2.94	1.93	83.70	20.05	13.05	74.57	24.08	27.04
5	18	62,634	10.26	2.04	1.43	75.75	13.75	10.26	64.21	20.65	26.12
10	28	40,265	7.57	1.67	0.98	59.40	11.35	6.50	46.77	15.82	25.16
15	38	29,670	5.36	1.43	0.67	43.90	9.65	4.20	33.55	12.93	24.35
20	48	23,488	3.75	1.30	0.62	31.50	8.90	4.00	24.05	10.77	23.54

Table 2: Results for experiments with the query set $ValQFixed(39)$ over the dataset **LANDSAT Korea** . The number of sensors $n = 200$; field size is $100m \times 100m$; the base station is located at (50m, 300m). The fixed extent v is 10% of the size of the domain for the dataset. The exact answer averaged over all queries for 60 rounds and 20 placements is 58.32. For NoA, packet size in bytes, $p = 10$ and for NA, $p = 2n + 8$. Both NoA and NA approaches always give exact results. For DWT, DFT and LS k is the size of the sketch and the size of the packets, $p = (2k + 8)$ bytes.

Method	p	LifeTime	MSE			RLE			REN		
NoA	10	12,472	0			0			0		
NA	408	865	0			0			0		
Other Aggregation methods (DWT/DFT/LS)											
k	p	LifeTime	MSE			RLE %			REN %		
			DWT	DFT	LS	DWT	DFT	LS	DWT	DFT	LS
1	10	37,749	57.92	7.48	7.48	99.44	13.78	13.78	93.36	29.65	29.65
3	14	27,925	55.88	6.64	5.80	96.50	12.14	10.45	85.26	27.57	29.04
5	18	20,901	52.73	5.56	3.78	91.84	10.11	6.85	79.45	26.57	28.26
10	28	13,398	45.05	4.47	2.30	80.05	8.08	4.10	67.54	24.85	27.65
15	38	10,487	38.04	3.95	1.41	68.92	7.16	2.47	57.88	23.52	27.24
20	48	7,945	31.22	3.72	1.26	57.72	6.73	2.19	49.38	22.39	26.85

DWT would be much worse if we were to estimate the top-k coefficients incrementally. More importantly, it can be seen that the aggregation techniques provide much higher lifetimes than NA and NoA. For the **Synthetic50** dataset (Table 1), with a sketch size of 5, LS achieved a mean lifetime of $62,634$ which is a 200% improvement over NoA and 500% improvement over NA and still had only 10.68% relative error. With a sketch size of 20, LS had a mean relative error of only 4% but a mean lifetime of 23,488 which is a 13% improvement over NoA. Similar results were observed for the LANDSAT datasets as well. For the (see

Table 3: Results for experiments with query set $ValQFixed(35)$ over the dataset **LANDSAT Miami** . The number of sensors $n = 200$; the field size is $100m \times 100m$; the base station is located at (50m, 300m). The fixed extent v is 10% of the size of the domain for the dataset. The exact answer averaged over all queries for 60 rounds and 20 placements is 56.03. For NoA, packet size in bytes, $p = 10$ and for NA, $p = 2n + 8$. Both NoA and NA approaches always give exact results. For DWT, DFT and LS k is the size of the sketch and the size of the packets, $p = (2k + 8)$ bytes.

Method	p	LifeTime	MSE			RLE			REN		
NoA	10	12,472	0			0			0		
NA	408	865	0			0			0		
Other Aggregation methods (DWT/DFT/LS)											
k	p	LifeTime	MSE			RLE %			REN %		
			DWT	DFT	LS	DWT	DFT	LS	DWT	DFT	LS
1	10	37,749	54.57	10.68	10.68	97.98	20.03	20.03	96.44	26.59	26.59
3	14	27,925	52.02	6.21	3.57	94.32	11.53	6.48	90.87	22.31	23.14
5	18	20,901	49.68	3.60	2.81	90.91	6.55	4.99	86.20	19.94	22.93
10	28	13,398	44.39	3.39	1.84	83.00	6.16	3.28	76.45	17.72	22.63
15	38	10,487	39.73	3.18	1.21	75.81	5.78	2.12	68.41	16.46	22.38
20	48	7,945	35.48	3.05	1.10	69.18	5.54	1.93	61.45	15.47	22.14

Tables 2 and 3), LS achieved 67.6% and 7.4% improvements in lifetimes for a small loss (of less than 10% relative error) in accuracy. Therefore it is evident that the LS method that we proposed indeed improves lifetime of the sensor network for a small loss in accuracy.

Experiment 2: The set up for the next set of experiments is same as above, except that we use a different query set $ValQRand(v, \sigma^2)$ for each round, where v is 10% of the range that the values that the sensors record and σ^2 is 4, 16 and 16 for the datasets **Synthetic50** , **LANDSAT Miami** and **LANDSAT Korea** respectively. For the LS method we use the $ValQFixed(v)$ set of queries for forming the sketching matrix. The results are reported in tables 4, 5 and 6. Note that although we use the sketching matrix for a different set of queries in the case of LS, we are still able to estimate answers to the original queries very well. For all the three datasets, we observe that there is less than 10% relative error when we use the LS method with a sketch size of only 5, and significantly improved lifetimes just as in the previous experiment. Therefore, it is clear that the LS sketching method works even for queries that are not originally used to prepare the sketching matrix.

Experiment 3: Finally we perform experiments to investigate how our approaches work for location range queries. For each placement over both the fields we consider location query sets $LocQFixed(16)$, $LocQFixed(30)$ and $LocQFixed(32)$ for **Synthetic50** ,**LANDSAT Miami** and **LANDSAT Korea** datasets respectively. For the LS method, we use the top–k eigenvectors of the matrix QQ^* as the sketching matrix. The results are summarized in tables 7, 8 and 9. Note that the accuracies achieved by LS are significantly better than those achieved by the DWT and DFT. For the **Synthetic50** dataset, LS had a mean relative error of 6.01% with a sketch size of 15. The MLT using this sketch size was 23,488 which is 13% an 125% better

Table 4: Results for experiments with the query set $ValQRand(10,4)$ over the dataset **Synthetic50** . The number of sensors $n = 50$; the field size is $50m \times 50m$; the base station is located at (25m, 150m). The exact answer averaged over all queries for 60 rounds and 20 placements is 16.02. For NoA, packet size in bytes, $p = 10$ and for NA, $p = 2n + 8$. Both NoA and NA approaches always give exact results. For DWT, DFT and LS k is the size of the sketch and the size of the packets, $p = (2k + 8)$ bytes.

Method	p	LifeTime	MSE			RLE			REN		
NoA	10	20,716	0			0			0		
NA	108	10,438	0			0			0		
Other Aggregation methods (DWT/DFT/LS)											
k	p	LifeTime	MSE			RLE %			REN %		
			DWT	DFT	LS	DWT	DFT	LS	DWT	DFT	LS
1	10	112,742	14.71	3.87	3.87	94.55	24.95	24.95	89.40	31.20	31.20
3	14	80,530	12.58	3.03	1.93	84.90	19.65	12.35	74.57	24.08	27.04
5	18	62,634	11.04	2.06	1.38	77.10	13.35	8.85	64.21	20.65	26.12
10	28	40,265	8.18	1.66	0.91	60.65	10.80	6.00	46.77	15.82	25.16
15	38	29,670	5.77	1.42	0.66	44.75	9.25	4.05	33.55	12.93	24.35
20	48	23,488	4.01	1.29	0.63	31.85	8.55	3.95	24.05	10.77	23.54

Table 5: Results for the query set with random queries $ValQRand(39,16)$ over the dataset **LANDSAT Korea** . The number of sensors $n = 200$. The field size is $100m \times 100m$; the base station is located at (50m, 300m). The exact answer averaged over all queries for 60 rounds and 20 placements is 58.65. For NoA, packet size in bytes, $p = 10$ and for NA, $p = 2n + 8$. Both NoA and NA approaches always give exact results. For DWT, DFT and LS k is the size of the sketch and the size of the packets, $p = (2k + 8)$ bytes.

Method	p	LifeTime	MSE			RLE			REN		
NoA	10	12,472	0			0			0		
NA	408	865	0			0			0		
Other Aggregation methods (DWT/DFT/LS)											
k	p	LifeTime	MSE			RLE %			REN %		
			DWT	DFT	LS	DWT	DFT	LS	DWT	DFT	LS
1	10	37,749	57.31	8.39	8.39	97.93	15.67	15.67	93.36	29.65	29.65
3	14	27,925	54.88	7.53	6.74	94.39	14.05	12.35	85.26	27.57	29.04
5	18	20,901	51.69	6.44	4.63	89.75	12.02	8.83	79.45	26.57	28.26
10	28	13,398	44.16	5.26	2.90	78.44	9.90	5.67	67.54	24.85	27.65
15	38	10,487	37.32	4.71	1.95	67.79	8.90	3.88	57.88	23.52	27.24
20	48	7,945	30.76	4.42	1.78	57.30	8.38	3.53	49.38	22.39	26.85

than the MLTs achieved by NoA and NA respectively. For the **LANDSAT Korea** and **LANDSAT Miami** datasets, LS achieves mean relative error of 16.71% and 11.85% respectively when the sketch size is 10, when the MLT is 13,398 that is 7.4% better than NoA. Therefore, from this experiment, we observe that LS achieve better estimates for the location queries than what DWT or DFT achieve and that too using small sketch sizes that gives improved lifetimes.

Table 6: Results for the query set with random queries $ValQRand(35, 16)$ over the dataset **LANDSAT Miami** . The number of sensors $n = 200$. The field size is $100m \times 100m$; the base station is located at (50m, 300m). The exact answer averaged over all queries for 60 rounds and 20 placements is 53.80. For NoA, packet size in bytes, $p = 10$ and for NA, $p = 2n + 8$. Both NoA and NA approaches always give exact results. For DWT, DFT and LS k is the size of the sketch and the size of the packets, $p = (2k + 8)$ bytes.

Method	p	LifeTime	MSE			RLE			REN		
NoA	10	12,472	0			0			0		
NA	408	865	0			0			0		
Other Aggregation methods (DWT/DFT/LS)											
k	p	LifeTime	MSE			RLE %			REN %		
			DWT	DFT	LS	DWT	DFT	LS	DWT	DFT	LS
1	10	37,749	52.48	10.53	10.53	98.17	21.06	21.06	96.44	26.59	26.59
3	14	27,925	50.16	6.15	3.53	94.84	12.24	6.93	90.87	22.31	22.14
5	18	20,901	48.03	3.57	2.74	91.71	7.01	5.20	86.20	19.94	22.93
10	28	13,398	43.15	3.35	1.80	84.38	6.58	3.44	76.45	17.72	22.63
15	38	10,487	38.82	3.14	1.20	77.64	6.16	2.26	68.41	16.46	22.38
20	48	7,945	34.85	3.01	1.10	71.39	5.89	2.07	61.45	15.47	22.14

Table 7: Results for the query set with location queries of fixed extent, $LocQFixed(16)$ over the dataset **Synthetic50** . The number of sensors $n = 50$; the field size is $50m \times 50m$; the base station is located at (25m, 150m). The exact answer averaged over all queries for 60 rounds and 20 placements is 68. For NoA, packet size in bytes, $p = 10$ and for NA, $p = 2n + 8$, where n is the number of sensors. Both NoA and NA approaches always give exact results. For DWT, DFT and LS k is the size of the sketch and the size of the packets, $p = (2k + 8)$ bytes.

Method	p	LifeTime	MSE			RLE			REN		
NoA	10	20,716	0			0			0		
NA	108	10,438	0			0			0		
Other Aggregation methods (DWT/DFT/LS)											
k	p	LifeTime	MSE			RLE %			REN %		
			DWT	DFT	LS	DWT	DFT	LS	DWT	DFT	LS
1	10	112,742	60.01	14.77	57.38	89.07	29.79	85.61	86.59	9.78	92.41
3	14	80,530	42.51	13.21	40.32	65.16	26.18	63.50	59.87	8.27	77.29
5	18	62,634	30.16	12.07	29.58	47.84	23.71	46.99	39.33	7.16	68.30
10	28	40,265	15.04	10.00	13.94	26.25	19.16	22.37	14.18	5.12	49.93
15	38	29,670	9.36	8.06	7.10	17.92	15.23	11.13	5.40	3.64	39.00
20	48	23,488	6.31	6.58	3.78	11.53	12.16	6.01	2.84	2.53	31.28

4 Conclusion

Data–fusion using in–network aggregation has become a core service provided in sensor networks. This is mainly because of the reduction in the amount of raw data to be communicated from the sensors to the base–station. From the database perspective, in–network aggregation provides a mechanism to evaluate aggregate queries. Query processing in sensor networks typically consists of two phases: query diffusion phase and the aggregation phase during which the query is evaluated by maintaining the aggregate for that query using

Table 8: Results for the query set with location queries of fixed extent, $LocQFixed(32)$ over the dataset **LANDSAT Korea** . The number of sensors $n = 200$. The field size is $100m \times 100m$; the base station is located at (50m, 300m). The exact answer averaged over all queries for 60 rounds and 20 placements is 1147. For NoA, packet size in bytes, $p = 10$ and for NA, $p = 2n + 8$, where n is the number of sensors. Both NoA and NA approaches always give exact results. For DWT, DFT and LS k is the size of the sketch and the size of the packets, $p = (2k + 8)$ bytes.

Method	p	LifeTime	MSE			RLE			REN		
NoA	10	12,472	0			0			0		
NA	408	865	0			0			0		
Other Aggregation methods (DWT/DFT/LS)											
k	p	LifeTime	MSE			RLE %			REN %		
			DWT	DFT	LS	DWT	DFT	LS	DWT	DFT	LS
1	10	37,749	957.34	465.07	872.75	81.58	68.94	72.55	87.01	28.82	93.66
3	14	27,925	628.33	447.64	497.61	51.65	66.23	44.35	63.55	27.25	80.75
5	18	20,901	448.34	428.01	348.13	48.01	63.11	32.13	43.61	25.96	73.69
10	28	13,398	418.84	384.44	161.59	61.80	56.64	16.71	25.25	23.32	58.38
15	38	10,487	375.50	349.12	106.37	55.11	51.28	11.27	22.06	21.16	51.33
20	48	7,945	337.25	320.09	81.43	49.23	46.70	8.43	19.47	19.32	48.58

Table 9: Results for the query set with location queries of fixed extent, $LocQFixed(30)$ over the dataset **LANDSAT Miami** . The number of sensors $n = 200$. The field size is $100m \times 100m$; the base station is located at (50m, 300m). The exact answer averaged over all queries for 60 rounds and 20 placements is 1446. For NoA, packet size in bytes, $p = 10$ and for NA, $p = 2n + 8$, where n is the number of sensors. Both NoA and NA approaches always give exact results. For DWT, DFT and LS k is the size of the sketch and the size of the packets, $p = (2k + 8)$ bytes.

Method	p	LifeTime	MSE			RLE			REN		
NoA	10	12,472	0			0			0		
NA	408	865	0			0			0		
Other Aggregation methods (DWT/DFT/LS)											
k	p	LifeTime	MSE			RLE %			REN %		
			DWT	DFT	LS	DWT	DFT	LS	DWT	DFT	LS
1	10	37,749	1250	294	919	86.23	21.17	67.18	85.90	14.72	85.38
3	14	27,925	826	277	503	55.44	20.30	37.01	57.83	13.95	71.16
5	18	20,901	475	266	329	31.11	19.47	25.33	33.27	13.34	62.17
10	28	13,398	264	244	159	19.27	17.82	11.85	12.76	12.06	49.55
15	38	10,487	235	223	100	17.25	16.31	7.46	10.95	10.98	43.34
20	48	7,945	212	207	75	15.62	15.13	5.67	9.55	10.05	40.32

in–network aggregation. While the queries can be answered accurately using this approach, it does not scale well with the number of queries.

Many monitoring applications need just approximate query results. In such applications, it is worthwhile to explore techniques that improve lifetime for a small loss in accuracy of the query results. In this paper we explore this idea. We take the approach taken in OnLine Analytical Processing (OLAP) for approximate query answering. The idea is to compute a summary data structure over the data from all the sensors using in–network aggregation, and answer all the queries in an approximate manner using this summary data structure at the base–station. The summary data structure depends on the kind of queries we wish to answer. In this paper we consider two kinds aggregate range queries: value range queries that count the number of sensors that record values in the given range, and location range queries that find the sum of values from sensors in a given rectangular area. We show how linear sketches can be used to answer such queries effectively and efficiently to get improved lifetimes for only a small loss in accuracy. Linear sketches have the advantage of being optimized for a given set of queries. We performed several experiments to compare the accuracy of queries as well as lifetime of the sensor networks. The proposed method of linear sketching achieves much better accuracy compared to results using DFT and DWT for significant gains in lifetimes compared to the naive aggregation and no aggregation schemes, in the case of both value range queries as well as location range queries.

References

[1] http://www.nnic.noaa.gov/socc/gallery.htm. Web page.

[2] Reference removed for double blinding. Technical report.

[3] Shivnath Babu and Jennifer Widom. Continuous queries over data streams. Technical report, Stanford University, 2001.

[4] J.H. Chang and L. Tassiulas. Energy Conserving Routing in Wireless Ad-hoc Networks. In *Proceedings of IEEE INFOCOM*, 2000.

[5] J.H. Chang and L. Tassiulas. Maximum Lifetime Routing in Wireless Sensor Networks. In *Proceedings of Advanced Telecommunications and Information Distribution Research Program, College Park, MD*, 2000.

[6] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proceedings of 20th IEEE International Conference on Data Engineering (ICDE)*, 2004.

[7] Philip B. Gibbons, Yossi Matias, and Viswanath Poosala. Aqua project white paper. Technical report, Information Sciences Research Center, Bell Laboratories, 1997.

[8] Phillip B. Gibbons and Yossi Matias. Synopsis data structures for massive data sets. *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science: Special Issue on External Memory Algorithms and Visualization*, A, 1999.

[9] Sudipto Guha, Piotr Indyk, S. Muthukrishnan, and Martin Strauss. Histogramming data streams with fast per-item processing. In *ICALP 2002*, pages 681–692.

[10] W. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocols for Wireless Microsensor Networks. In *Proceedings of Hawaiian International Conference on Systems Science*, 2000.

[11] Joseph M. Hellerstein, Wei Hong, Samuel R. Madden, and Kyle Stanek. Beyond Average: Towards Sophisticated Sensing with Queries. In *Proceedings of 2nd International Workshop on Information Processing in Sensor Networks (IPSN)*, 2003.

[12] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of 6th ACM/IEEE Mobicom Conference*, 2000.

[13] K. Kalpakis, K. Dasgupta, and P. Namjoshi. Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks. In *Proceedings of IEEE International Conference on Networking*, 2002.

[14] Konstantinos Kalpakis, Koustuv Dasgupta, and Parag Namjoshi. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks*, 42(6):697–716, August 2003.

[15] Konstantinos Kalpakis, Koustuv Dasgupta, and Parag Namjoshi. Improving the lifetime of sensor networks via intelligent selection of data aggregation trees. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'03)*, 2003.

[16] B. Krishnamachari, D. Estrin, and S. Wicker. The Impact of Data Aggregation in Wireless Sensor Networks. In *Proceedings of International Workshop on Distributed Event-Based Systems*, 2002.

[17] S. Lindsey and C. S. Raghavendra. PEGASIS: Power Efficient GAthering in Sensor Information Systems. In *Proceedings of IEEE Aerospace Conference*, 2002.

[18] S. Lindsey, C. S. Raghavendra, and K. Sivalingam. Data Gathering in Sensor Networks using the Energy*Delay Metric. In *Proceedings of the IPDPS Workshop on Issues in Wireless Networks and Mobile Computing*, 2001.

[19] S. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. In *Proceedings of 4th IEEE Workshop on Mobile Computing and Systems Applications*, 2002.

[20] Sam Madden and Michael J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *ICDE*, 2002.

[21] Samuel. R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the ACM Symposium on Operating System Design and Implementation (OSDI)*, December 2002.

[22] Nitin Thaper, Sudipto Guha, Piotr Indyk, and Nick Koudas. Dynamic multidimensional histograms. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 428–439. ACM Press, 2002.

[23] Yong Yao and Johannes E. Gehrke. Query processing in sensor networks. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, 2003.

Contents