Exploiting Spatio-temporal Correlations for Data Processing in Sensor Networks

Antonios Deligiannakis[1] and Yannis Kotidis[2]

[1] University of Athens
adeli@di.uoa.gr
[2] Athens University of Economics and Business
kotidis@aueb.gr

Abstract. Recent advances in microelectronics have made feasible the deployment of sensor networks for a variety of monitoring and surveillance tasks. In such tasks the state of the network is evaluated either at regular intervals at a base-station, which constitutes a centralized location where the data collected by the sensor nodes can be collected and processed, or continuously through the use of, potentially multiple, continuous queries. In order to increase the network lifetime, multiple techniques have been proposed in order to reduce the data transmitted in the network, since the data communication often constitutes the main source of energy drain in sensor networks. In this work we discuss several data reduction techniques that can be applied for energy-efficient query processing in sensor network applications. All of our proposed techniques seek to identify and take into account the characteristics of the collected data. Depending on the nature of the monitoring application at hand, the targeted data characteristics may range from simply monitoring the variance of a node's measurements to identifying spatio-temporal correlations amongst the values collected by the sensor nodes.

1 Introduction

Recent advances in wireless technologies and microelectronics have made feasible, both from a technological as well as an economical point of view, the deployment of densely distributed sensor networks [61]. Although today's sensor nodes have relatively small processing and storage capabilities, driven by the economy of scale, it is already observed that both are increasing at a rate similar to Moore's law.

In applications where sensors are powered by small batteries and replacing them is either too expensive or impossible (i.e., sensors thrown over a hostile environment), designing energy efficient protocols is essential to increase the lifetime of the sensor network. Since radio operation is by far the biggest factor of energy drain in sensor nodes [18], minimizing the number of transmissions is vital in data-centric applications. Even in the case when sensor nodes are attached to larger devices with ample power supply, reducing bandwidth consumption may still be important due to the wireless, multi-hop nature of communication and the short-range radios usually installed in the nodes.

S. Nittel, A. Labrinidis, and A. Stefanidis (Eds.): GSN 2006, LNCS 4540, pp. 45–65, 2008.

Data-centric applications thus need to devise novel dissemination processes for minimizing the number of messages exchanged amongst the nodes. Nevertheless, in densely distributed sensor networks there is an abundance of information that can be collected. In order to minimize the volume of the transmitted data, we can apply two well known ideas *aggregation* and *approximation* in order to exploit spatio-temporal correlations in the readings obtained by the nodes in the network.

In-network aggregation is more suitable for exploratory, continuous queries that need to obtain a live estimate of some (aggregated) quantity. For example, sensors deployed in a metropolitan area can be used to obtain estimates on the number of observed vehicles. Temperature sensors in a warehouse can be used to keep track of the average and maximum temperature for each floor of the building. Often, aggregated readings over a large number of sensors nodes show little variance, providing a great opportunity for reducing the number of (re)transmissions by the nodes when individual measurements change only slightly (as in temperature readings) or changes in measurements of neighboring nodes effectively cancel out (as in vehicle tracking applications).

Approximation techniques, in the form of lossy data compression are more suitable for the collection of historical data through long-term queries. As an example, consider sensors dispersed in a wild forest, collecting meteorological measurements (such as pressure, humidity, temperature) for the purpose of obtaining a long term historical record and building models on the observed eco-system [2,42,67]. Each sensor generates a multi-valued data feed and often substantial compression can be achieved by exploiting natural correlations among these feeds (such as in case of pressure and humidity measurements). In such cases, sensor nodes are mostly "silent" (thus preserving energy) and periodically process and transmit large batches of their measurements to the monitoring station for further processing and archiving.

While the preferred method of data reduction, either by aggregation or by approximation, of the underlying measurements can be decided based on the application needs, there is a lot of room for optimization at the network level as well. Sensor networks are inherently redundant; a typical deployment uses a lot of redundant sensor nodes to cope with node or link failures [4]. Thus, extracting measurements from all nodes in the network for the purpose of answering a posed query may be both extremely expensive and unnecessary. In a data-centric network, nodes can coordinate with their neighbors and elect a small set of *representative nodes* among themselves, using a localized, data-driven bidding process [35]. These representative nodes constitute a *network snapshot* that can, in turn, answer posed queries while reducing substantially the energy consumption in the network. These nodes are also used as an alternative means of answering a posed query when nodes and network links fail, thus providing unambiguous data access to the applications.

This article proceeds as follows. Section 2 provides a brief overview of the characteristics of sensor nodes. Section 3 presents our Self-Based Regression (SBR) algorithm for the compression of historical measurements in sensor

network applications, while Section 4 presents our framework for the approximate evaluation of continuous aggregate queries. Section 5 presents our techniques for creating a *network snapshot* that can be used to efficiently evaluate queries about the observed values of the sensor nodes, while Section 6 presents some related work in the area of sensor networks. Finally, Section 7 contains concluding remarks and future directions.

2 Characteristics of Sensor Nodes

Depending on the targeted application, sensor nodes with widely different characteristics may be used. Even though the processing and memory capabilities of sensor nodes are still limited, in recent years they have increased at a rate similar to Moore's law. On the other hand, the amount of energy stored in the batteries used in such nodes has exhibited a mere 2-3% annual growth. Since replacing the sensor batteries may be very expensive, and sometimes impossible due to their unattended deployment, unless the sensors are attached to and powered by a larger unit, designing energy-efficient protocols is essential to increase the lifetime of the sensor network.

The actual energy consumption by each sensor node depends on its current state. In general, each sensor node can be in one of the following states:

- *low-duty cycle*, where the sensor is in sleep mode and a minimal amount of energy is consumed.
- *idle listening*, where the sensor node is listening for possible data intended for it.
- *processing*, where the sensor node performs computation based on its obtained measurements and its received data.
- *receiving/transmitting*, where the sensor node either receives or transmits data or control messages.

The cost of processing can be significant but is generally much lower than the cost of transmission. For example, in the Berkeley MICA nodes sending one bit of data costs as much energy as 1,000 CPU instructions [41]. For long-distance radios, the transmission cost dominates the receiving and idle listening costs. For short-range radios, these costs are comparable. For instance, in the Berkeley MICA2 motes the power consumption ratio of transmitting/receiving at 433MHz with RF signal power of 1mW is 1.41:1 [64], while this ratio can become even larger than 3:1 for the same type of sensor when the radio transmission power is increased [54]. To increase the lifetime of the network, some common goals of sensor network applications are (in order of importance) to maximize the time when a node is in a low-duty cycle, to reduce the amount of transmitted and received data, and to reduce the idle listening time. We note here that reducing the size of the transmitted data results in multiple benefits, since this also corresponds to a reduction of not only control messages, but also leads to fewer message collisions and retransmissions. Moreover, nodes that refrain from transmitting messages may switch to the low-duty cycle mode faster, therefore further reducing their energy drain.

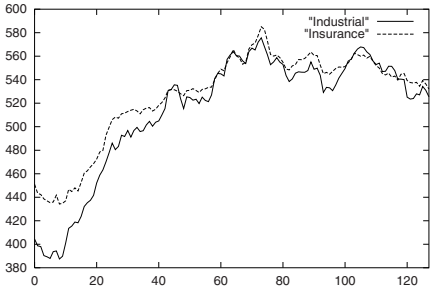

Fig. 1. Example of two correlated signals (Stock Market)

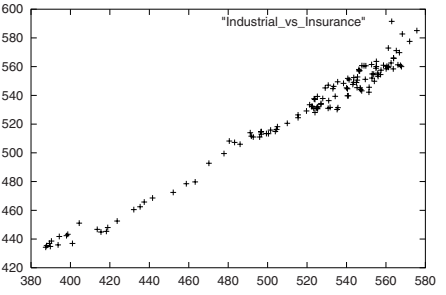

Fig. 2. XY scatter plot of Industrial (X axis) vs Insurance (Y axis)

3 A Lossy Compression Framework for Historical Measurements

Many real signals observed by the sensors, such as temperature, dew-point, pressure etc. are naturally correlated. The same is often true in other domains. For instance, stock market indexes or foreign currencies often exhibit strong correlations. In Figure 1 we plot the average Industrial and Insurance indexes from the New York stock market for 128 consecutive days. Both indexes show similar fluctuations, a clear sign of strong correlation. Figure 2 depicts a XY scatter plot of the same values. This plot is created by pairing values of the Industrial (X-coordinate) and Insurance (Y-coordinate) indexes of the same day and plotting these points in a two-dimensional plane. The strong correlation among these values makes most points lie on a straight line. This observation suggests the following compression scheme, inspired from regression theory. Assuming that the Industrial index (call it $\boldsymbol{X}$) is given to us in a time-series of 128 values, we can approximate the other time-series (Insurance: $\boldsymbol{Y}$) as $\boldsymbol{Y'} = a * \boldsymbol{X} + b$. The coefficients a and b are determined by the condition that the sum of the square residuals, or equivalently the L_2 error norm $||\boldsymbol{Y'} - \boldsymbol{Y}||_2$, is minimized. This is nothing more than standard linear regression. However, unlike previous methods, we will not attempt to approximate each time-series independently using regression. In Figure 1 we see that the series themselves are not linear, i.e., they would be poorly approximated with a linear model. Instead, we will use regression to approximate piece-wise correlations of each series to a base signal $\boldsymbol{X}$ that we will choose accordingly. In the example of Figure 2, the base signal can be the Industrial index ($\boldsymbol{X}$) and the approximation of the Insurance index will be just two values (a, b). In practice the base signal will be much smaller than the complete time series, since it only needs to capture the "important" trends of the target signal $\boldsymbol{Y}$. For instance, in case $\boldsymbol{Y}$ is periodic, a sample of the period would suffice.

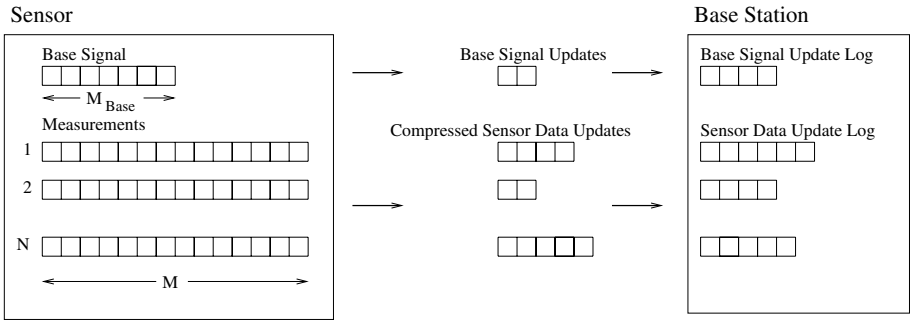

Fig. 3. Transfer of approximate data values and of the base signal from each sensor to the base station

3.1 The SBR Framework

In the general case, each sensor monitors N distinct quantities $\boldsymbol{Y_i}$, $1 \leq i \leq N$. Without loss of generality we assume that measurements are sampled with the same rate. When enough data is collected (for instance, when the sensor memory buffers become full), the latest NxM values are processed and each row i (of length M) is approximated by a much smaller set of B_i values, i.e. $B_i \ll M$. The resulting "compressed" representation, of total size equal to $\sum_{i=1}^{N} B_i$, is then transmitted to the base station. The base station maintains the data in this compact representation by appending the latest "chunk" to a log file. A separate file exists for each sensor that is in contact with the base station. This process is illustrated in Figure 3. Each sensor allocates a small amount of memory of size M_{base} for what we call the *base signal*. This is a compact ordered collection of values of prominent features that we extract from the recorded values and are used as a base reference in the approximate representation that is transmitted to the base station. The data values that the sensor transmits to the base station are encoded using the in-memory values of the base signal at the time of the transmission. The base signal may be updated at each transmission to ensure that it will be able to capture newly observed data features and that the obtained approximation will be of good quality. When such updates occur, they are transmitted along with the data values and appended in a special log file that is unique for each sensor.

The Self-Based Regression algorithm (SBR) breaks the data intervals $\boldsymbol{Y_i}$ into smaller data segments

$$I_i[k..l] = (Y_i[k], \ldots Y_i[l])$$

and then pairs each one to an interval of the base signal of equal length. As discussed below, the base signal is simply the concatenation of several intervals of the same length W extracted from the data. The data interval I_i is shifted over the base signal and at each position s we compute the regression parameters for the approximation

$$\hat{I}_i[j] = a \times X[s + j - k] + b, k \leq j \leq l$$

and retain the shift value $s = s^*$ that minimizes the sum-squared error of the approximation. The algorithm starts with a single data interval for each row of the collected data (Y_i). In each iteration, the interval with the largest error in the approximation is selected and divided in two halves. The compressed representation of a data interval $I_i[k..l]$ consists of four values: the shift position s^* that minimizes the error of the approximation, the two regression parameters a,b and the start of the data interval k in Y_i. The base station will sort the intervals based on their start position and, thus, there is no need to transmit their ending position. Given a target budget B (size of compressed data) we can use at most $B/4$ intervals using this representation.

3.2 Base Signal Construction

We can think of the base signal as a dictionary of features used to describe the data values. The richer the pool of features we store in the base signal the better the approximation. On the other hand, these features have to be (i) kept in the memory of the sensor to be used as a reference by the data-reduction algorithm and (ii) sent to the base station in order for it to be able to reconstruct the values. Thus, for a target bandwidth constraint B (number of values that can be transmitted), performing more insert and update operations on the base signal implies less bandwidth remaining for approximating the data values, and, thus, fewer data intervals that can be obtained from the recursive process described above.

We can avoid the need of transmitting the base signal by agreeing a-priori on a set of functions that will be used in the regression process. For instance, a set of cosine functions (as in the Distinct Cosine Transform) can be used for constructing a "virtual" base signal that does not need to be communicated. Similarly, using the identity function $X[i] = i$ reduces the compression algorithm to standard linear regression of each data interval. However, such an approach makes assumptions that may not hold for the data at hand. In [11] we have proposed a process for generating the base signal from the data values. The key idea is to break the measurements into intervals of the same length W. Each interval (termed *candidate base interval*) is assigned a score based on the reduction in the error of the approximation obtained by adding the interval to the base signal. Using a greedy algorithm we can select the top-few candidate intervals, up to the amount of available memory M_{base}. Then a binary-search process is used to eventually decide how many of those candidate intervals need to be retained.

The search space is illustrated in Figure 4 for three real data sets, discussed in [11]. The figure plots the error of only the initial transmission as the size of the base signal is varied, manually, from 1 to 30 intervals. We further show the selection of the binary-search process. For presentation purposes, the errors for each data set have been divided by the error of the approximation when using just one interval. We notice that initially, by adding more candidate intervals

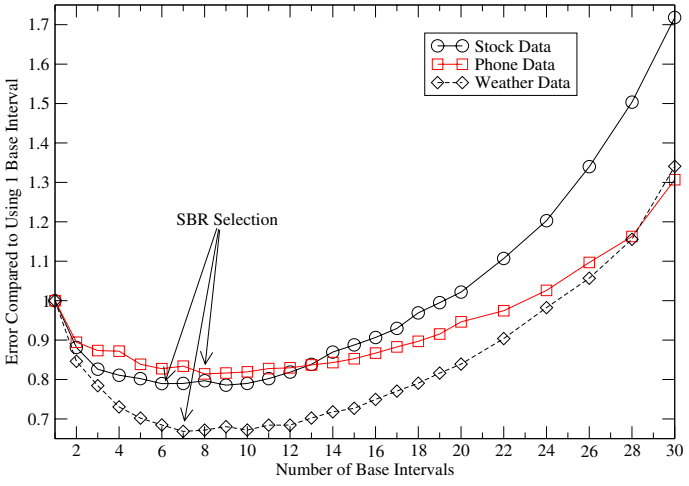

Fig. 4. SSE error vs base signal size

to the base signal the error of the approximation is reduced. However, after a point, adding more intervals that need to be transmitted to the base station leaves insufficient budget for the recursive process that splits the data, and thus, the error of the approximation is eventually increased.

3.3 Analysis and Evaluation

For a data set containing $n = N \times M$ measurements to approximate the complete SBR algorithm takes $O(n^{1.5})$ time and requires linear space, while its running time scales linearly to the size of both the transmitted data and the base signal. The algorithm is invoked periodically, when enough data has been collected. Moreover, our work [11] has demonstrated that, after the initial transmissions, the base signal is already of good quality and few intervals are inserted in it. This provides us with the choice not to update the base signal in many subsequent invocations, thus reducing the algorithm's running time, in these cases, to only a linear dependency on n.

To illustrate the accuracy achieved by the SBR algorithm against standard approximation techniques such as Wavelets, Histograms and the Discreet Cosine Transform (DCT), we used in [11], among other data sets, a weather data set that contains the air temperature, dewpoint temperature, wind speed, wind peak, solar irradiance and relative humidity weather measurements obtained from a station in the university of Washington, and for the year 2002. For this data set we selected the first 40,960 records and then split the data measurements of each signal into ten files of 4,096 values each, in order to simulate multiple transmissions. We then varied the compression ratio (size of the transmitted data over the data size n) from 5% to 50% and present in Table 1 the total sum

Table 1. Total SSE Varying the Compression Ratio for the Weather Data Set

Compression Ratio	Weather Data (245,760 total values)			
	SBR	Wavelets	DCT	Histograms
5%	317,238	519,303	8,703,192	7,661,293
10%	103,633	200,501	4,923,294	3,375,518
15%	54,219	125,449	3,515,698	2,219,533
20%	30,946	87,118	2,643,229	1,471,421
25%	18,600	63,105	2,198,455	946,735
30%	11,558	46,833	1,598,451	594,644
35%	7,161	35,275	1,366,211	410,208
40%	4,603	26,824	1,112,117	288,127
45%	2,964	20,502	905,422	236,947
50%	1,861	15,762	768,568	160,079

squared error achieved by all techniques. In all cases, SBR produces significantly more accurate results than the other approximations.

3.4 Extensions

In [14] we present extensions to the basic SBR scheme that we describe here. These extensions allow the nodes to organize in groups based on an adaptation of the HEED protocol [65] and elect within each group a group leader that instruments the execution of each SBR instance in the nodes of its group and also handles the final transmission of the compressed data to the base station. This form of localized processing allows the nodes to exploit spatial correlations and results in a reduction of the error of the approximation by at least an order of magnitude compared to the case when nodes individually compress and transmit their data.

4 Approximate In-Network Data Aggregation

The data aggregation process in sensor networks consists of several steps. First, the posed query is disseminated through the network, in search of nodes collecting data relevant to the query. Each sensor then selects one of the nodes through which it received the announcement as its *parent node*. The resulting structure is often referred to as the *aggregation tree*. Non-leaf nodes of that tree aggregate the values of their children before transmitting the aggregate result to their parents. In [40], after the aggregation tree has been created, the nodes carefully schedule the periods when they transmit and receive data. The idea is for a parent node to be listening for values from its child nodes within specific intervals of each *epoch* and then transmit upwards a single partial aggregate for the whole subtree.

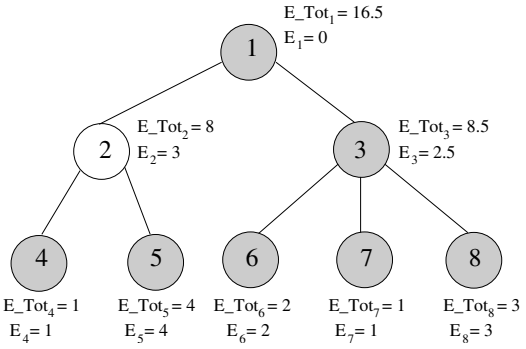

Fig. 5. Error Filters on Aggregation Tree

In order to limit the number of transmitted messages and, thus, the energy consumption in sensor networks during the computation of continuous aggregate queries, our algorithms install error filters on the nodes of the aggregation tree. Each node N_i transmits the partial aggregate that it computes at each epoch for its subtree only if this value deviates by more than the maximum error E_i of the node's filter from the last transmitted partial aggregate. This method allows nodes to refrain from transmitting messages about small changes in the value of their partial aggregate. The E_i values determine the maximum deviation E_Tot_i of the reported from the true aggregate value at each node. For example, for the SUM aggregate, this deviation at the monitoring node is upper bounded (ignoring message losses) by: $\sum_i E_i$.

A sample aggregation tree is depicted in Figure 5. As our work [12] has demonstrated, allowing a small error in the reported aggregate can lead to dramatic bandwidth (and thus energy) savings. The challenge is, of course, given the maximum error tolerated by the monitoring node, to calculate and periodically adjust the node filters in order to minimize the bandwidth consumption.

4.1 Algorithmic Challenges

When designing adaptive algorithms for in-network data aggregation in sensor networks, one has to keep in mind several challenges/goals. First, communicating *individual* node statistics is very expensive, since this information cannot be aggregated inside the tree, and may outweigh the benefits of approximate data aggregation, namely the reduction in the size of the transmitted data. Thus, our algorithm should not try to estimate the number of messages generated by each node's transmissions, since this depends on where this message is aggregated with messages from other nodes. Second, the error budget should be distributed to the nodes that are expected to reduce their bandwidth consumption the most by such a process. This benefit depends on neither the magnitude of the partial aggregate values of the node nor the node's number of transmissions over a period, but on the magnitude of the changes on the calculated partial aggregate.

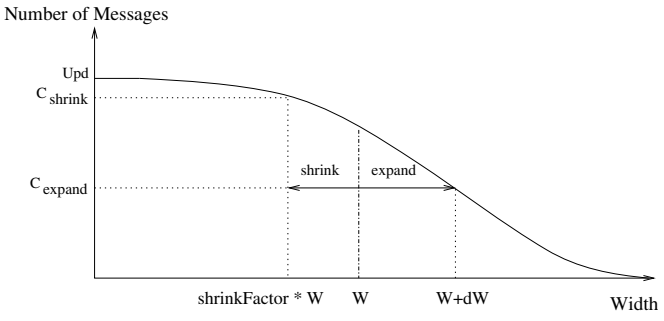

Fig. 6. Potential Gain of a Node

Isolating nodes with large variance on their partial aggregate and redistributing their error to other nodes is crucial for the effectiveness of our algorithm [12]. Finally, in the case of nodes where the transmitted differences from their children often result in small changes on the partial aggregate value, our algorithm should be able to identify this fact. We deal with this latter challenge by applying the error filters on the calculated partial aggregate values and not on each node's individual measurements. For the first two challenges, we collect a set of easily computed and composable statistics at each node. These statistics are used for the periodic adjustment of the error filters.

4.2 Algorithm Overview

Every Upd epochs all nodes shrink the widths of their filters by a shrinking factor $0 < shrinkFactor < 1$. After this process, the monitoring node has an error budget of size $E\_Global \times (1 - shrinkFactor)$, where E_Global is the maximum error of the application, that it can redistribute recursively to the nodes of the network. Each node, between two consecutive update periods, calculates its *potential gain* as follows: At each epoch the node keeps track of the number of transmissions C_{shrink} that it would have performed with the default (smaller) filter at the next update period of width $shrinkFactor \times W_i$, where $W_i = 2 \times E_i$. The node also calculates the corresponding number of transmissions C_{expand} with a larger filter of width $W_i + dW$ and sets its potential gain to $Gain_i = C_{shrink} - C_{expand}$. This process is illustrated in Figure 6. The *cumulative gain* of a node's subtree is then calculated as the sum of the cumulative gains of the node's children and the node's potential gain. This requires only the transmission of the cumulative gains (a single value for each node) at the last epoch before the new update period. The available error budget is then distributed top-down proportionally, at each node, to each subtree's cumulative gain. In this process, nodes that exhibit large variance in their partial aggregate values will exhibit small potential gains and, thus, their error will gradually shrink and be redistributed to nodes that will benefit from an increase in their error filter.

We note here that the dual problem of minimizing the application maximum error given a bandwidth or energy constraint is also very interesting. This

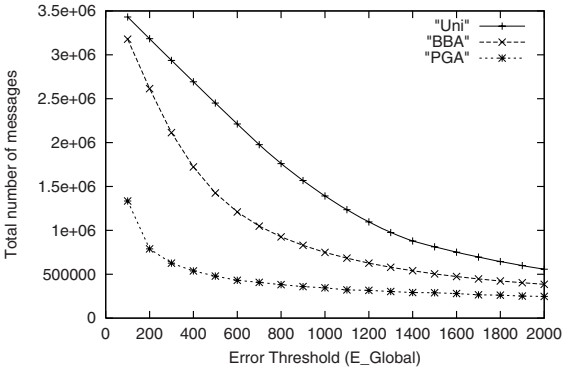

Fig. 7. Reduction in messages for synthetic data set

problem is discussed in [13] and is more complicated, though, because the controlled quantity at each node (the width of its error filter) is different from the monitored quantity (the bandwidth/energy consumption) and the bandwidth needs to be carefully computed, monitored and then disseminated amongst the sensor nodes.

4.3 Experimental Evaluation

We used a balanced aggregation tree with 5 levels and a fan-out of 4 (341 nodes overall), where all the nodes collected measurements relevant to the query. In the first experiment, the measurements of all the nodes followed a random walk pattern, each with a randomly assigned step size in the range $(0 \ldots 2]$. To capture the scenario where nodes update their measurements with different frequencies, 20% of the nodes update their measurements at each epoch, while the remaining sensors make a random step with a fixed probability of 1% during an epoch. In Figure 7 we plot the total number of messages in the network (y-axis) for 40,000 epochs when varying the error constraint E_Global from 100 to 2,000 (8% is terms of relative error). Depending on E_Global, our PGA (Potential Gains Adjustment) algorithm results in up to 4.8 times fewer messages than an adaptation of the algorithm proposed by Olston in [45], that we termed BBA, and up to 6.4 times fewer messages than a uniform allocation policy (Uni) where the error is partitioned uniformly amongst all nodes. These differences arise from the ability of PGA to place, judiciously, filters on passive intermediate sensor nodes and exploit negative correlations on their subtree based on the computed potential gains. Algorithm BBA may also place filters on the intermediate nodes (when the residual mode is used) but the selection of the widths of the filters based on the burden scores of the nodes was typically not especially successful in our experiments.

5 Design of Data-Centric Sensor Networks

Sensor networks are inherently dynamic. Such networks must adapt to a wide variety of challenges imposed by the uncontrolled environment in which they operate. As nodes become cheaper to manufacture and operate, one way of addressing the challenges imposed on unattended networks is redundancy [4]. Redundant nodes ensure network coverage in regions with non-uniform communication density due to environmental dynamics. Redundancy further increases the amount of data that can be mined in large-scale networks.

Writing data-driven applications in such a dynamic environment can be daunting. The major challenge is to design localized algorithms that will perform most of the processing in the network itself in order to reduce traffic and, thus, preserve energy. Instead of designing database applications that need to hassle with low-level networking details, we envision the use of data-centric networks that allow transparent access to the collected measurements in a unified way. For instance, when queried nodes fail, the network should self-configure to use redundant stand-by nodes as in [18], under the condition that the new nodes contain fairly similar measurements, where similarity needs to be quantified in an application-meaningful way [35]. This can be achieved using a localized mode of operation in which nodes coordinate with their neighbors and elect a small set of *representative nodes* among themselves. Such a set of representatives, termed *network snapshot* [35], has many advantages.

- The location and measurements of the representative nodes provide a picture of the value distribution in the network. By choosing an error metric (such as sum-squared or relative error) and using different threshold values to express similarity amongst the sensor node measurements we can obtain different snapshots of the network at different resolutions, depending on the error threshold used.
- The network snapshot can be used for answering user queries in a more energy-efficient way. The data reduction techniques that we discussed in Sections 3 and 4 aim at reducing the flow of data in the network by either suppressing update messages or compressing long data streams. The network snapshot is an orthogonal optimization that can further reduce energy drain during query processing by reducing the number of nodes that need to respond to user queries. When a user query can tolerate a small error in the computation, the network can use the representative nodes and compile a quick answer from only a fraction of the nodes that a normal query execution would require. We call such queries *snapshot queries*.
- An elected representative node can take over for another node in the vicinity that may have failed or is temporarily out of reach. Because selection of representatives is quantitative this allows for a more accurate computation than when representatives are selected based only on proximity.
- A localized computation of representative nodes can react quickly to changes in the network. For instance, nodes (including the representatives) may fail at random. It is important that the network can self-heal in the case of

node-failures or some other catastrophic events. In a data-driven mode of operation, we are also interested in changes in the behavior of a node. In such case the network should re-configure and revise the selected representatives, when necessary. What is important is that, as we demonstrate in [35], these computations can be performed in the network with only a small number (up to six) of exchanged messages among the nodes.

5.1 Snapshot Overview

A sensor node N_i maintains a data model for capturing the distribution of values of the measurements of its neighbors. This is achieved by snooping (with a small probability) values broadcast by its neighbor node N_j in response to a query or, by using periodic announcements sent by N_j. One may devise different data models, with varying degrees of complexity, for this process. In [11,35] we have proposed modeling the correlations amongst the measurements of the nodes using linear regression. Regression models are simple both in terms of space and time complexity. As is demonstrated in [35], our algorithms can operate when the available memory for storing these models in the sensor is as small as a few bytes. Furthermore, by modeling the correlations amongst the values of the nodes we avoid making assumptions on the distribution of the data values collected by the sensors that may not hold in practice. The only assumption made is that values of neighboring nodes are to some extent correlated. This is true for many applications of interest like collection of meteorological data, acoustic data etc, as discussed in Section 3.

Using the data model it maintains, sensor node N_i provides an estimate $\hat{x}_j$ of the measurement x_j of its neighbor N_j. Given an error metric $d()$ and a threshold value T, node N_i can *represent* node N_j if $d(x_j, \hat{x}_j) \leq T$. Function $d()$ is provided by the application. Some common choices include (i) relative error: $d(x_j, \hat{x}_j) = \frac{|x_j - \hat{x}_j|}{\max(s, |x_j|)}$, where $s > 0$ is a sanity bound for the case x_j=0, (ii) absolute error: $d(x_j, \hat{x}_j) = |x_j - \hat{x}_j|$ and (iii) sum-squared error: $d(x_j, \hat{x}_j) = (x_j - \hat{x}_j)^2$.

Through a localized election process (see [35] for details) the nodes in the network pick a set of representative nodes of size n_1. Depending on the threshold value T, the error metric and the actual measurements on the sensors, n_1 can be significantly smaller than the number of nodes in the network. An example of this process is demonstrated in Figure 8 where the representatives for a simulated network of 100 nodes are shown. Dark nodes in the Figure are representative nodes. There are lines drawn from a node N_i to a node N_j that N_i represents. Nodes with no lines attached to them represent themselves (the default choice).

An aggregate computation like SUM can be handled by the representative nodes that will in-turn provide estimates on the nodes N_j they represent using their models. Another scenario is to use the representative of a node on an aggregate or direct query, when that node is out-of-reach because of some unexpected technical problem or due to severe energy constraints. Thus, query processing can take advantage of the unambiguous data access provided by the network. Of course, one can ignore the layer of representatives and access the sensors directly,

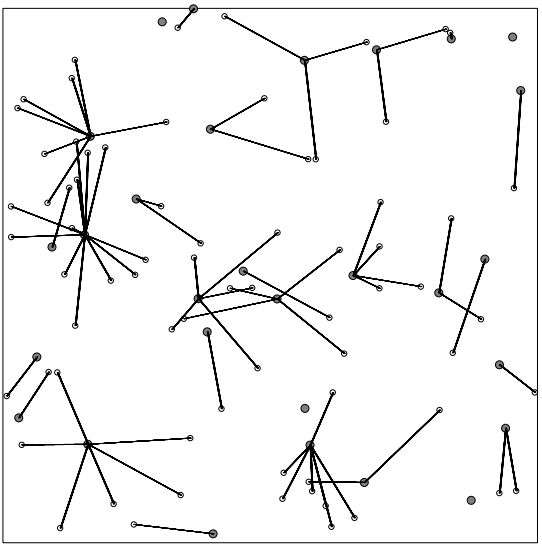

Fig. 8. Example of Network Snapshot

at the penalty of (i) draining more energy, since a lot more nodes will have be to accessed for the same query and (ii) having to handle *within the application* node failures, redundancy etc.

The selection of representatives is not static but is being revised overtime in an adaptive fashion. An obvious cause is the failure of a representative node. In other cases, the model built by N_i to describe x_j might get outdated or fail, due to some unexpected change in the data distribution. In either case, the network will self-heal using the following simple protocol. Node N_j periodically sends a heart-beat message to its representative N_i including its current measurement. If N_i does not respond, or its estimate $\hat{x}_j$ for x_j is not accurate ($d(x_j, \hat{x}_j) > T$) then N_j initiates a local re-evaluation process inviting candidate representatives from its neighborhood, leading to the selection of a new representative node (that may be itself). This heart-beat message is also used by N_i to fine-tune its model of N_j.

Under an unreliable communication protocol it is possible that this process may lead to *spurious* representatives. For instance node N_i may never hear the messages sent by node N_j due to an obstacle in their direct path. It may thus assume that it still represents node N_j while the network has elected another representative. This can be detected and corrected by having time-stamps describing the time that a node N_i was elected as the representative of N_j and using the latest representative based on these time-stamps. In TinyOS nodes have an external clock that is used for synchronization with their neighbors [40]. In lack of properly synchronized clocks among the sensor nodes, one can use a global counter like the epoch-id of a continuous query. This filtering and self-correction is performed by the network, transparently from the application.

5.2 Examples of Snapshot Queries

Recent proposals [40,63] have suggested the use of SQL for data acquisition and processing. The obvious advantage of using a declarative language is greater flexibility over hand-coded programs that are pre-installed at the sensor nodes [41]. In addition embedded database systems like TinyDB can provide energy-based query optimization because of their tight integration with the node's operations.

Basic queries in sensor networks consist of a SELECT-FROM-WHERE clause (supporting joins and aggregation). For instance, in our running example of collecting weather data a typical query may be

```
SELECT loc, temperature
FROM sensors
WHERE loc in SHOUTH_EAST_QUANDRANT
SAMPLE INTERVAL 1sec for 5min
USE SNAPSHOT
```

This is an example of a drill-through query, sampling temperature readings every 1 second and lasting 5 minutes. For this example we assume that each node has a unique location-id *loc* and that nodes are location-aware, being able to process the spatial filter "in SHOUTH_EAST_QUANDRANT". Location can be obtained using inexpensive GPS receivers embedded in the nodes, or by proper coordination among the nodes, using techniques like those proposed in [50,52]. Location is a significant attribute of a node in an unattended system. For many applications like habitat monitoring, spatial filters may be the most common predicate.

The new extension presented in the query above is the USE SNAPSHOT clause, denoting that the query may be answered by the representative set of nodes. An example of an aggregate query that is using the snapshot for computing the average and maximum temperature readings in the same area is given below

```
SELECT avg(temperature), max(temperature)
FROM sensors
WHERE loc in SHOUTH_EAST_QUANDRANT
SAMPLE INTERVAL 1sec for 5min
USE SNAPSHOT
```

5.3 Evidence of Savings During Snapshot Queries

We used a simulated network of 100 sensor nodes, randomly placed in a $[0 \ldots 1) \times [0 \ldots 1)$ two-dimensional area. For each node, we generated values following a random walk pattern, each with a randomly assigned step size in the range $(0 \ldots 1]$. The initial value of each node was chosen uniformly in range $[0 \ldots 1000)$. We then randomly partitioned the nodes into K classes. Nodes belonging to the same class i were making a random step (upwards or downwards) with the same

Table 2. Reduction in number of nodes participating in a spatial snapshot query

	K=1		K=100	
	Transmission Range		Transmission Range	
Query Range	0.2	0.7	0.2	0.7
1%	11%	29%	3%	7%
10%	38%	77%	16%	24%
50%	52%	91%	23%	49%

probability $P_{move}[i]$. These probabilities were chosen uniformly in range $[0.2 \ldots 1]$ (we excluded values less than 0.2 to make data more volatile).

We tested aggregate queries over random parts of the network. For each query a *sink* node was chosen randomly. Then, using the flooding mechanism described in [40] an aggregation tree was formed, rooted at the sink node. The sensor nodes N_i whose measures were aggregated using that tree, were chosen using spatial predicate "$location_i$ *in* $[x - \frac{W}{2}, x + \frac{W}{2}] \times [y - \frac{W}{2}, y + \frac{W}{2}]$", where (x, y) is a random point in the $[0 \ldots 1) \times [0 \ldots 1)$ plane.

We created a random set of 200 such queries and executed each query in the set twice: once as a regular query and once as a snapshot query. We counted the number of nodes participated in each execution, denoted as $N_{regular}$ and $N_{snapshot}$ respectively. In Table 2 we show the savings $\frac{N_{regular} - N_{snapshot}}{N_{regular}}$ provided on the average by the snapshot queries (the error threshold was one). We note that when snapshot queries are used, a non-representative node may still be used for routing the aggregate and this is included in the numbers shown. We made two runs, one with a single class and another when each node was on a class of its own (K=100). We varied the size W of the range queries as shown in the table. We further tested two transmission ranges for the nodes. The shorter transmission range results in more representatives and taller aggregation trees, as more hops are required to reach the sink node. In Table 2 we can see that snapshot queries provide substantial savings in terms of the number of nodes participating in a query, especially on large spatial queries. For all runs, the aggregation tree was created using the vanilla method of [10,40]. One can modify the protocol to favor (when applicable) representative nodes for routing the messages. This will result in further reduction in the number of sensor nodes used during snapshot queries than those presented in Table 2.

6 Related Work

In recent years, there has been a significant body of work in the area of sensor networks. For instance, the networking aspects of wireless sensor nodes is a topic that has intrigued the networking community. Because of the unattended operation of sensor networks, nodes must be able to co-operate to perform the task at hand. Some of the most important topics addressed include network

self-configuration [4,35,65], discovery [18,27] and computation of energy-efficient data routing paths [6,28,39,40,55,57,65].

In the database community there are ongoing projects for infusing database primitives in the operations of these networks. For instance, TinyDB [41] and Cougar [63] have suggested the use of SQL for data acquisition and processing. The obvious advantage of using a declarative language is greater flexibility over hand-coded programs that are pre-installed at the sensor nodes [41]. In-network data aggregation is another topic that has created a flurry of proposals [12,16,19,30,40,53,63]. The main idea is to build an aggregation tree, which partial results will follow. With proper synchronization [40], non-leaf nodes of the tree aggregate the values of their children before transmitting a single aggregate result to their parents, thus substantially reducing the flow of messages in the network. Alternative, gossip-based techniques have also been investigated in [3,33]. In [10,35] the authors have also looked at the problems of packet loss and node failures during data processing. Recently, proposals for combining data modeling with data acquisition in order to help reduce the cost of query processing have been suggested [17,35,37]. For example, [17,37] build probabilistic models of the observed data and then use these models to probe the sensors for their measurements in a limited amount of epochs, depending on the confidence of the constructed model. Distributed storage management is another topic that brings together the networking and database communities [16,21,51].

Many of these fundamental techniques have been devised to support event-based monitoring applications. For example, in animal tracking, an event such as the presence of an animal can be determined by matching the sensor readings to stored patterns [31]. The authors of [62] propose an event detection mechanism based on matching the contour maps of in-network sensory data distributions. In [47], kernel-based techniques are used to detect abnormal behavior in sensor readings. In [26] the authors describe the implementation of a real system based on Mica2 motes for surveillance of moving vehicles. In [36] a framework for computing user-defined events that are in proximity is presented.

Query processing in sensor networks has some connection with the work on continuous queries in data streams [7,29,44,59,66]. The work of [46] studies the trade-off between precision and performance when querying replicated, cached data. In [45] the users register continuous queries with strict precision constraints at a central *stream processor*, which, in turn installs filters at the remote data sources. These filters adapt to changes in the streams to minimize update rates. Online algorithms for minimizing the update cost while the query can be answered within an error bound are presented in [34]. The authors of [9] study a probabilistic query evaluation method that places appropriate confidence in the query answer to quantify the *uncertainty* of the recorded data values.

There is a vast related literature on approximate processing techniques. The AQUA project explored sampling-based techniques for building *synopses* and using them to provide approximate answers at a fraction of the time that a real answer would require [22]. Histograms are used by query optimizers to estimate the selectivity of queries, and recently in tools for providing fast approximate

answers to queries [23,24,32,48,49,58]. Wavelets are a mathematical tool for the hierarchical decomposition of functions, with applications in image and signal processing [56]. More recently, Wavelets have been applied successfully in answering range-sum aggregate queries over data cubes [60], in selectivity estimation [43] and in approximate query processing [5,15,20,25]. The Discrete Cosine Transform (DCT) [1] constitutes the basis of the *mpeg* encoding algorithm and has also been used to construct compressed multidimensional histograms [38]. Linear regression has been recently used in [8] for on-line multidimensional analysis of data streams.

7 Conclusions and Future Directions

We have described several techniques for the reduction of the transmitted data in several sensor network applications, ranging from the communication of historical measurements to answering approximate aggregate continuous and snapshot queries. While these techniques aim to prolong the lifetime of the network, there are several issues that need to be additionally addressed. Little work has been done on the optimization of multiple concurrent continuous queries over sensor networks. The work of Olston et al. in [45] may provide some helpful solutions in this area. Moreover, in the presence of nodes with different transmission frequencies, as in the case of approximate aggregate query processing, several communication and synchronization algorithms may need to be revisited [63]. For example, the selection of the aggregation tree is often performed by assuming equal frequency of transmissions by all nodes. However, it might be more beneficial to prevent nodes that exhibit large variance in their measurements from appearing in lower levels of the tree, since such nodes often trigger transmissions on their ancestors as well. Such optimizations may lead to even larger energy savings.

References

1. Ahmed, N., Natarakan, T., Rao, K.R.: Discrete cosine transform. IEEE Trans. on Computers C-23 (1974)
2. Ailamaki, A., Faloutsos, C., Fischbeck, P.S., Small, M.J., Van Briesen, J.: An environmental sensor network to determine drinking water quality and security. SIGMOD Record 32(4), 47–52 (2003)
3. Bawa, M., Garcia-Molina, H., Gionis, A., Motwani, R.: Estimating Aggregates on a Peer-to-Peer Network. Technical report, Stanford (2003)
4. Cerpa, A., Estrin, D.: ASCENT: Adaptive Self-Configuring sEnsor Network Topologies. In: INFOCOM (2002)
5. Chakrabarti, K., Garofalakis, M., Rastogi, R., Shim, K.: Approximate Query Processing Using Wavelets. In: Proceedings of the 26th VLDB Conference (2000)
6. Chang, J.-H., Tassiulas, L.: Energy Conserving Routing in Wireless Ad-hoc Networks. In: INFOCOM (2000)
7. Chen, J., Dewitt, D.J., Tian, F., Wang, Y.: NiagaraCQ: A Scalable Continuous Query System for Internet Databases. In: Proceedings of ACM SIGMOD Conference (2000)

8. Chen, Y., Dong, G., Han, J., Wah, B.W., Wang, J.: Multi-Dimensional Regression Analysis of Time-Series Data Streams. In: Proceedings of VLDB (2002)
9. Cheng, R., Kalashnikov, D.V., Prabhakar, S.: Evaluating Probabilistic Queries over Imprecise Data. In: Proceedings of ACM SIGMOD Conference (2003)
10. Considine, J., Li, F., Kollios, G., Byers, J.: Approximate Aggregation Techniques for Sensor Databases. In: ICDE (2004)
11. Deligiannakis, A., Kotidis, Y., Roussopoulos, N.: Compressing Historical Information in Sensor Networks. In: Proceedings of ACM SIGMOD Conference (2004)
12. Deligiannakis, A., Kotidis, Y., Roussopoulos, N.: Hierarchical in-Network Data Aggregation with Quality Guarantees. In: Lindner, W., Mesiti, M., Türker, C., Tzitzikas, Y., Vakali, A.I. (eds.) EDBT 2004. LNCS, vol. 3268. Springer, Heidelberg (2004)
13. Deligiannakis, A., Kotidis, Y., Roussopoulos, N.: Bandwidth Constrained Queries in Sensor Networks. The VLDB Journal (2007)
14. Deligiannakis, A., Kotidis, Y., Roussopoulos, N.: Dissemination of Compressed Historical Information in Sensor Networks. The VLDB Journal (2007)
15. Deligiannakis, A., Roussopoulos, N.: Extended Wavelets for Multiple Measures. In: Proceedings of SIGMOD Conference, pp. 229–240 (2003)
16. Demers, A., Gehrke, J., Rajaraman, R., Trigoni, N., Yao, Y.: The Cougar Project: A Work In Progress Report. SIGMOD Record 32(4), 53–59 (2003)
17. Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J.M., Hong, W.: Model-Driven Data Acquisition in Sensor Networks. In: Proceedings of VLDB (2004)
18. Estrin, D., Govindan, R., Heidermann, J., Kumar, S.: Next Century Challenges: Scalable Coordination in Sensor Networks. In: MobiCOM (1999)
19. Ganesan, D., Estrin, D., Heidermann, J.: DIMENSIONS: Why do we need a new Data Handling architecture for Sensor Networks? In: HotNets-I (2002)
20. Garofalakis, M., Gibbons, P.B.: Probabilistic Wavelet Synopses. ACM Trans. Database Syst. 29(1), 43–90 (2004)
21. Ghose, A., Grossklags, J., Chuang, J.: Resilient Data-Centric Storage in Wireless Ad-Hoc Sensor Networks. In: Mobile Data Management (2003)
22. Gibbons, P.B., Matias, Y.: New Sampling-Based Summary Statistics for Improving Approximate Query Answers. In: Proceedings ACM SIGMOD International Conference on Management of Data, Seattle, Washington, pp. 331–342 (June 1998)
23. Gilbert, A., Guha, S., Indyk, P., Kotidis, Y., Muthukrishnan, S., Strauss, M.: Fast, Small-Space Algorithms for Approximate Histogram Maintenance. In: ACM STOC (2002)
24. Gilbert, A., Kotidis, Y., Muthukrishnan, S., Strauss, M.: Optimal and Approximate Computation of Summary Statistics for Range Aggregates. In: ACM PODS, pp. 227–236 (2001)
25. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: One-Pass Wavelet Decompositions of Data Streams. Trans. Knowl. Data Eng. 15(3), 541–554 (2003)
26. He, T., Krishnamurthy, S., Stankovic, J., Abdelzaher, T., Luo, L., Stoleru, R., Yan, T., Gu, L., Hui, J., Krogh, B.: An Energy-Efficient Surveillance System Using Wireless Sensor Networks. In: MobiSys. (2004)
27. Heidermann, J., Silva, F., Intanagonwiwat, C., Govindanand, R., Estrin, D., Ganesan, D.: Building Efficient Wireless Sensor Networks with Low-Level Naming. In: SOSP (2001)
28. Heinzelman, W., Chandrakasan, A., Balakrishnan, H.: Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In: Hawaii Conference on System Sciences (2000)

29. Hellerstein, J.M., Franklin, M.J., Chandrasekaran, S., Descpande, A., Hildrum, K., Madden, S., Raman, V., Shah, M.A.: Adaptive Query Processing: Technology in Evolution. IEEE DE Bulletin 23 (2000)
30. Intanagonwiwat, C., Estrin, D., Govindan, R., Heidermann, J.: Impact of Network Density on Data Aggregation in Wireless Sensor Networks. In: ICDCS (2002)
31. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In: MOBICOM. (2000)
32. Ioannidis, Y.E., Poosala, V.: Histogram-Based Approximation of Set-Valued Query Answers. In: Proceedings of the 25th VLDB Conference (2000)
33. Kempe, D., Dobra, A., Gehrke, J.: Gossip-Based Computation of Aggregate Information. In: Proceedings of FOCS (2003)
34. Khanna, S., Tan, W.C.: On Computing Functions with Uncertainty. In: Proceedings of ACM PODS Conference (2001)
35. Kotidis, Y.: Snapshot Queries: Towards Data-Centric Sensor Networks. In: Proceedings of ICDE (2005)
36. Kotidis, Y.: Processing Promixity Queries in Sensor Networks. In: Proceedings of DMSN (2006)
37. Lazaridis, I., Mehrotra, S.: Approximate Selection Queries over Imprecise Data. In: ICDE (2004)
38. Lee, J., Kim, D., Chung, C.: Multi-dimensional Selectivity Estimation Using Compressed Histogram Information. In: Proceedings of ACM SIGMOD Conference (1999)
39. Lindsey, S., Raghavendra, C.S.: Pegasis: Power-Efficient Gathering in Sensor Information Systems. In: IEEE Aerospace Conference (2002)
40. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tag: A Tiny Aggregation Service for ad hoc Sensor Networks. In: OSDI Conf. (2002)
41. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: The Design of an Acquisitional Query processor for Sensor Networks. In: Proceedings of ACM SIGMOD Conference (2003)
42. Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., Anderson, J.: Wireless Sensor Networks for Habitat Monitoring. In: WSNA 2002, pp. 88–97 (2002)
43. Matias, Y., Vitter, J.S., Wang, M.: Wavelet-Based Histograms for Selectivity Estimation. In: Proceedings of ACM SIGMOD Conference (1998)
44. Motwani, R., Widom, J., Arasu, A., Babcock, B., Babu, S., Datar, M., Manku, G., Olston, C., Rosenstein, J., Varma, R.: Query Processing, Resource Management, and Approximation in a Data Stream Management System. In: Proceedings of CIDR (2003)
45. Olston, C., Jiang, J., Widom, J.: Adaptive Filters for Continuous Queries over Distributed Data Streams. In: Proceedings of ACM SIGMOD Conference (2003)
46. Olston, C., Widom, J.: Offering a Precision-Performance Tradeoff for Aggregation Queries over Replicated Data. In: Proceedings of VLDB (2000)
47. Palpanas, T., Papadopoulos, D., Kalogeraki, V., Gunopulos, D.: Distributed Deviation Detection in Sensor Networks. SIGMOD Rec. 32(4) (2003)
48. Poosala, V., Ioannidis, Y.E.: Selectivity Estimation Without the Attribute Value Independence Assumption. In: Proceedings of the 23th VLDB Conference (1997)
49. Poosala, V., Ioannidis, Y.E., Haas, P.J., Shekita, E.J.: Improved Histograms for Selectivity Estimation of Range Predicates. In: Proceedings of ACM SIGMOD Conference (1996)
50. Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The Cricket Location-Support System. In: MOBICOM (2000)

51. Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R., Shenker, S.: GHT: a Geographic Hash Table for Data-Centric Storage. In: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (2002)
52. Savarese, C., Rabaey, J.M., Beutel, J.: Locationing in Distributed Ad-hoc Wireless Sensor Networks. In: ICASSP (2001)
53. Sharaf, A., Beaver, J., Labrinidis, A., Chrysanthis, P.: Balancing Energy Efficiency and Quality of Aggregate Data in Sensor Networks. VLDB Journal (2004)
54. Shnayder, V., Hempstead, M., Chen, B., Allen, G.W., Welsh, M.: Simulating the Power Consumption of Large-Scale Sensor Network Applications. In: Sensys. (2004)
55. Singh, S., Woo, M., Raghavendra, C.S.: Power-Aware Routing in Mobile Ad Hoc Networks. In: ACM/IEEE International Conference on Mobile Computing and Networking (1998)
56. Stollnitz, E.J., DeRose, T.D., Salesin, D.H.: Wavelets for Computer Graphics - Theory and Applications. Morgan Kaufmann Publishers, Inc., San Francisco (1996)
57. Tan, H.O., Korpeoglu, I.: Power Efficient Data Gathering and Aggregation in Wireless Sensor Networks. SIGMOD Record 32(4) (2003)
58. Thaper, N., Guha, S., Indyk, P., Koudas, N.: Dynamic Multidimensional Histograms. In: Proceedings of ACM SIGMOD Conference (2002)
59. Viglas, S.D., Naughton, J.F.: Rate-based Query Optimization for Streaming Information Sources. In: Proceedings of ACM SIGMOD Conference (2002)
60. Vitter, J.S., Wang, M.: Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. In: Proceedings of ACM SIGMOD Conference (1999)
61. Warneke, B., Last, M., Liebowitz, B., Pister, K.S.J.: Smart Dust: Communicating with a Cubic-Millimeter Computer. IEEE Computer 34(1), 44–51 (2001)
62. Xue, W., Luo, Q., Chen, L., Liu, Y.: Contour Map Matching for Event Detection in Sensor Networks. In: SIGMOD (2006)
63. Yao, Y., Gehrke, J.: The Cougar Approach to In-Network Query Processing in Sensor Networks. SIGMOD Record 31(3), 9–18 (2002)
64. Ye, W., Heidermann, J.: Medium Access Control in Wireless Sensor Networks. Technical report, USC/ISI (2003)
65. Younis, O., Fahmy, S.: HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks. IEEE Transactions on Mobile Computing 3(4) (2004)
66. Zdonik, S.B., Stonebraker, M., Cherniack, M., Cetintemel, U., Balazinska, M., Balakrishnan, H.: The Aurora and Medusa Projects. IEEE DE Bulletin (2003)
67. Zeinalipour-Yazti, D., Neema, S., Gunopulos, D., Kalogeraki, V., Najjar, W.: Data Acquision in Sensor Networks with Large Memories. In: IEEE International Workshop on Networking Meets Databases, Tokyo, Japan (April 2005)