Available online at www.sciencedirect.com
ScienceDirect
Signal Processing 87 (2007) 2965–2977
SIGNAL PROCESSING
www.elsevier.com/locate/sigpro

A new deterministic data aggregation method for wireless sensor networks ☆

Hüseyin Akcan*, Hervé Brönnimann

Computer & Information Science Department, Polytechnic University, Brooklyn, NY 11201, USA

Received 31 August 2006; received in revised form 28 February 2007; accepted 15 May 2007
Available online 29 May 2007

Abstract

The processing capabilities of wireless sensor nodes enable to aggregate redundant data to limit total data flow over the network. The main property of a good aggregation algorithm is to extract the most representative data by using minimum resources. From this point of view, sampling is a promising aggregation method, that acts as surrogate for the whole data, and once extracted can be used to answer multiple kinds of queries (such as AVG, MEDIAN, SUM, COUNT, etc.), at no extra cost to the sensor network. Additionally, sampling also preserves correlations between attributes of multi-dimensional data, which is quite valuable for further data mining. In this paper, we propose a novel, distributed, weighted sampling algorithm to sample sensor network data and compare to an existing random sampling algorithm, which is the only algorithm to work in this kind of setting. We perform popular queries to evaluate our algorithm on a real world data set, which covers climate data in the US for the past 100 years. During testing, we focus on issues such as sample quality, network longevity, energy and communication costs.

Keywords: Sampling; Data reduction; In-network data aggregation; Data streams; Sensor networks

1. Introduction

In this paper, we focus on sampling data from a set of sensor nodes linked by a network and consider the problem of extracting the sample from the network to a central DBMS and later querying it to find answers to conditions inside the sensor network. These queries may be based on snapshots, or may be continuous. The data collected by the sensors is usually highly redundant, and thus one need not collect and process all of it, approximate query results are usually sufficient. Hence, substantial savings may be obtained by either aggregating or processing the data in-network, or by obtaining a much smaller but representative sample which can then be processed by a centralized more powerful unit.

One particular kind of data we focus on is the multi-dimensional count data that arises fairly often from market basket data, census-like applications, or in the context of sensor networks, from monitoring several parameters of an environment. For instance, Mainwaring et al. [1] engineer Mica Weather Boards to record temperature, humidity,

☆Research of the authors has been supported by NSF CAREER GRANT CCR-0133599.
*Corresponding author. Tel.: +1 718 2603378.
E-mail addresses: hakcan01@cis.poly.edu (H. Akcan), hbr@poly.edu (H. Brönnimann).

0165-1684/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.sigpro.2007.05.007

and barometric pressure. Additional components like photo-resistor and infrared thermopile sensors can join forces to infer other parameters such as cloud cover, altitude or wind speed. Levon [2] is developing a biosensor array for detecting biological warfare agents; this array can report and monitor the presence of a number of chemical compounds, and the data is inherently multi-dimensional as we are not only interested in each parameter but also in their interaction. In this context, one may desire to investigate possible correlations, to discover that certain combination of values occur more often than others if certain conditions are met (association rules), or to construct a data cube (or iceberg cube if only the most significant statistical data is desired). Note that outliers are also of potential interest, especially to act as triggers; they can be monitored using completely different techniques, so we do not consider those here, rather we focus on extracting the most significant statistical trends.

We contend that collecting and maintaining a representative sample of the data from the sensor network is a promising data aggregation solution because samples, unlike other synopsis structures, are general purpose and can be used as a surrogate for the (expensive) network [3]. In other domains, sampling has long been used to answer many aggregation queries approximately [4] such as MEDIAN, AVG, COUNT, and MODE. It also has been successfully used to speed up data mining tasks such as finding correlations and association rules [5–9], and iceberg cubes [5,10]. In the context of sensor networks, using sampling as a surrogate can be most appropriate for this kind of global analysis, because the whole data must be available to find correlations, and gathering the whole data in a centralized unit requires prohibitive communication costs. Tailored aggregation procedures have better accuracy, but each requires its own communication costs, while the sample is computed and maintained once and enables any number of a wide range of queries out-of-network with still reasonable accuracy at no cost to the network. Moreover, an approximate answer often suffices and more precise results can be obtained otherwise by 'drilling down' in the network in an OLAP fashion.

Although random sampling is an easy and intuitive way to answer approximate queries, random deviations in the sampling process, however, make random sampling somewhat less precise and unpredictable. In the context of transactional data sets, a simple deterministic procedure (EASE [6], refined in Biased-L2 [5]) produces samples with errors consistently an order of magnitude better than that of a random sample. In the context of sensor networks, we find that this translates into order-of-magnitude communication and energy savings, namely, for an equivalent RMS error, our deterministic sample is much smaller thus requires lower energy costs (including extra communication requirements of the algorithm). Alternatively, for the same resource spending, one gets a much more accurate picture.

We also would like to mention a few applications of our sampling scheme. By integrating spatial regions as categorical attributes, our sample will try to be representative in space as well as for the measurement. That is, it will have the same spatial distribution as the original set of sensors, at least at the resolution of regions. Note that the regions may overlap. The regions of a hierarchical subdivision such as quadtrees or hierarchical uniform grids may be desirable to ensure scale-independence, therefore regions adapted to a natural decomposition such as counties for census data may sometimes be more appropriate. In monitoring homeland security chemical threats, being able to detect the emergence and geographic spread of a hostile chemical compound [2] involves combining geographic information with that of the biosensor array to ensure that environmental conditions are accurately sampled. For this to be accurate, the sampling rates (both temporal and spatial) must be adjustable at the node level, which can easily be achieved by tweaking the weights in our weighted sampling approach.

The rest of the paper is organized as follows: we give an overview of the previous work in Section 2. In Section 3, we present an arbitrary aggregation structure for sampling, discuss the motivation for weighted sampling in this kind of aggregation structures and finally present our deterministic sampling algorithm. In Section 4 we introduce the real world climate data set we used in our simulations, and then experimentally study our algorithm by comparing to existing sampling algorithms, while focusing on issues such as sample quality, energy and communication costs. Finally, we conclude in Section 5 with both discussion and future work.

Our contributions: Our main contributions in this paper are

- We propose a novel deterministic weighted sampling (DWS) logrithm as a new aggregation

method for sensor network data. To the best of our knowledge, this is the first distributed deterministic sampling algorithm for sensor networks.
- Our algorithm weights the samples and adjusts the weights dynamically, which enables to work on networks organized in any arbitrary topology.
- In our DWS algorithm, data aggregation via sampling is done on all (participating) nodes, which equally distributes sampling work over the network, and prevents any node from being a bottleneck (both CPU and communication based).

2. Related work

In this section, we give an overview of the previous work and state the relations and differences compared to our work.

In a preliminary version [11], we present the basic algorithm and experiments with the random data set. In this version, we further extend on the previous work both in terms of the algorithm details and experiments.

In [5], Biased-L2 is presented, which improves on EASE [6,12] to deterministically sample streaming count data. Both algorithms sample by introducing penalty functions using the support of each item. Each penalty function is minimized when the item frequency over the sample equals that over the whole data set, and increases sharply when item is under- or over-sampled. EASE reduces the number of records in the sample by applying halving steps on the sample, while Biased-L2 is a single pass algorithm that gives a final decision for adding the record to the sample or not by examining the record only once. EASE and Biased-L2 are designed to work on centralized database settings, where data is stored in a database or comes from a single stream, so they are not directly applicable to sensor network settings, where data is distributed and network topology is unknown. In our paper, we propose a novel deterministic sampling algorithm that uses similar ideas to Biased-L2, but extended to handle weights and distributed sources, essential for sensor network data extraction.

The research on stream processing is highly relevant to our work. Stream database systems aim to process data streams as fast as possible while avoiding the need to store the entire data. In most cases, this results in approximate answers to the queries. Aqua [4] uses an approximation engine to quickly answer typical aggregate queries while incrementally refining the result with the actual database results. Borealis [13] is a high-performance, low-latency distributed stream processing engine. In addition to typical snapshot queries, STREAM [14] queries in continuous fashion, which naturally adapts to stream data. Continuous queries are also used in sensor network databases TinyDB [15] and COUGAR [16].

Recently, data aggregation in sensor networks attracted great attention from the research community [16–19]. In sensor networks, where the in-network processing of various aggregate queries is paramount, data aggregation inside the network could drastically reduce the communication cost (consequently, prolong the battery life) and ensure the desired bounds on the quality of data. Madden et al. [19] propose an effective aggregation tree (TAG-tree), which works on well-defined epochs, and reduces the communication by using an optimum tree structure. We also use a tree generation algorithm similar to TAG-tree, where iterative broadcast from sink to the rest of the network is used to create a tree that covers the whole network. COUGAR [16] also creates aggregation trees similar to TAG. Sharaf et al. [20] propose more efficient ways of aggregation by exploiting group by queries, which works on top of TAG and COUGAR aggregation. In addition to tree aggregation, cluster-based aggregation approaches have also been investigated in LEACH [17,21] and PEGASIS [18]. In LEACH [17], randomly selected cluster heads perform aggregation, and communicate directly with the base station to reduce energy use, while in LEACH-C [21] the base station broadcasts the cluster head assignments, further improving the energy use of the network. PEGASIS [18] selects the cluster head by organizing nodes in a chain. Using only one cluster head at a time conserves energy at the expense of introducing latency issues.

Although the tree structure is optimal for communication, it is not robust enough for sensor networks. The robustness issue led researchers to propose a DAG architecture instead of an aggregation tree. Addressing the duplication problem of the DAG architecture, Considine et al. [22] and Nath et al. [23] independently propose using duplicate-insensitive sketches for data aggregation in sensor networks. While Considine et al. focus only on efficiently counting SUM aggregates, Nath et al. give a general framework of defining and identifying

order and duplicate insensitive (ODI) synopsis, including a uniform random sample (DIR sample as we call it in this paper). In our paper, we compare our algorithm to DIR, which computes a uniform random sample of the sensor data on any arbitrary hierarchical network structure, using an optimum number of messages. The algorithm initially assigns random weights to data values. To generate a sample of size s, each node selects the sample values having the biggest s weights. The selected sample values are propagated up the hierarchy to the sink. Since during each sampling stage, the top s weighted samples are selected, after the final stage, a uniform sample of size s is obtained. Similar to this approach, our algorithm also uses a deterministic way to generate s samples for each node, thus effectively distributing sampling work equally over every node.

Manjhi et al. [24] by combining the tree and multi-path aggregation structures, propose a tributary-delta structure. Tree and multi-path structures coexist in this new structure, and convert to each other, so the tributary delta can behave as effective as a tree and as robust as a multi-path, depending on the need and network topology. Shrivastava et al. [25] propose an aggregation technique for medians, specifically for sensor networks. Greenwald and Khanna [26] extend their space-efficient order statistics algorithms for sensor networks.

Compression is used recently as an aggregation method for sensor networks. Lazaridis and Mehrotra [27] propose using a piecewise constant approximation algorithm to compress the time series with quality guarantees. Since the method proposed is lossy compression for single dimension time series only, applying it would not keep the valuable correlation information in multi-dimensional data. We assume our data is multi-dimensional (such as temperature, barometer, etc.) and we are mostly interested in the correlation of these dimensions. Deligiannakis et al. [28] propose extracting a base signal from the data and further using piecewise linear regression to compress the original signal, using the base signal. The algorithm explicitly uses the correlation between multiple dimensions of data (if exists) to increase the effect of compression. Multi-dimensional lossy compression methods (that keep correlations) and lossless compression methods are complementary to our work, since the samples we create are subsets of the original data, and any method to compress the data can be applied to our samples too. In this paper, to focus on the original behavior of the algorithms, we present the algorithms without the added benefit of compression.

In their work, Heinzelman et al. [17] and Chen et al. [29] present energy formulas for wireless transmission and receive. According to [29], the energy usage during idle::receive::transmit is, respectively, 1::1.2::1.7. We also use the same energy formulas in our evaluations.

3. Distributed deterministic sampling

In this section, we first present the notation essential to understand our algorithm. Later, we discuss the shortcomings of the existing deterministic sampling algorithms for arbitrary sensor network topologies, and present the necessities for using weights in the sampling algorithm. Finally, we present our DWS algorithm, which is designed to run on distributed environments. The simplicity and effectiveness of DWS is most appropriate to run on resource-restraint hardware such as sensor nodes.

3.1. Notation

Let D denote the database of interest, $d = |D|$ the number of records, S a deterministic sample drawn from D, and $s = |S|$ its number of records. Each record consists of a set of items. We denote by I the set of all items that appear in D, by m the total number of such items, and by size(t) the number of items appearing in a single record $t \in D$. We let T_{avg} denote the average number of items in a record, so that dT_{avg} denotes the total size of the database (as counted by a complete item per record enumeration).

For a set T of records and an itemset $A \subseteq I$, we let $n(A;T)$ be the number of records in T that contain A and $|T|$ the total number of records in T. Then the support of A in T is given by $f(A;T) = n(A;T)/|T|$. In particular, $f(A;D) = n(A;D)/|D|$ and $f(A;S) = n(A;S)/|S|$. The distance between D and S with respect to the 1-itemset frequencies can be computed via the L2-norm (also called 'root-mean-square,' or RMS),

$$\text{RMS}(D,S) = \sqrt{\sum_{A \in I} (f(A,D) - f(A,S))^2}. \quad (1)$$

Our goal is to select a sample S of D of size $\alpha|D|$ (where $0 < \alpha \leqslant 1$ is the *sampling rate*) that minimizes the distance $\text{RMS}(D,S)$. Although L2-norm is our choice of distance metric in this paper, there are

other possible notions of distance, for instance the *discrepancy* (L_∞-norm) can be used if the maximum deviation is important.

Finally, to relate the notation to our sensor data, we assume that nodes have multiple sensors on them (such as temperature, barometer, etc.). Each reading from a sensor corresponds to an item. A total reading from a node includes all the sensor values for a particular time. The collection of all the sensor readings correspond to records, and a whole record is transferred between nodes during aggregation.

3.2. Aggregation data structure

In sensor networks we assume that the data is distributed over the nodes, hence each node x holds a subset D_x of D such that $\cup_x D_x = D$ (In the extreme case, each node holds a single value, which may be updated over time). We also assume that an aggregation tree structure is already available for our algorithms. We do not require any specific property on the tree, other than its connectedness (it must cover all the nodes). Once the aggregation tree is built, sensor nodes, starting from the leaves of the tree, create a sample of size $s = \alpha_x|D_x|$ from their data D_x, and forward these samples to their parents. Nodes in the middle levels of the tree wait until they gather samples from all their children (or for a timeout), and then do sampling on all the gathered data, including their own, to create a sample of size s. Since the desired sample size s is known, and D_x depends on the number of children, each node adjusts its sampling rate α_x accordingly. This sampling scheme is similar to the DIR [23], and maintains a sample of size s for every node in the network.

3.3. Motivation for weighted sampling

One important challenge for us in distributed sampling is that, since we do not enforce any structure on the aggregation tree, the tree topology changes with the underlying sensor network topology, and each node in the tree has an arbitrary number of children. In particular, the sampling rate on each node varies depending on the number of children. Simply gathering all the children's samples in a big chunk of data and deterministically or randomly sampling over this chunk would introduce misrepresentations in the sample of a node. Namely, the sample values of nodes closer to the sink in the tree would have more chance to appear in the final sample. An example aggregation tree clearly showing this situation is presented in Fig. 1. The top node has three samples of size s, the one coming from the left subtree, the right subtree and finally its own data. In this case, each sample has a different weight and a regular sampling by ignoring these weights would give each of these three samples a $\frac{1}{3}$ chance to appear in the final sample. This means $\frac{1}{3}$ chance for the top node's data, $\frac{1}{6}$ chance for each left-subtree node's data and $\frac{1}{9}$ chance for each right-subtree node's data. As we would like to give each node's data an equal $\frac{1}{6}$ chance to be represented in

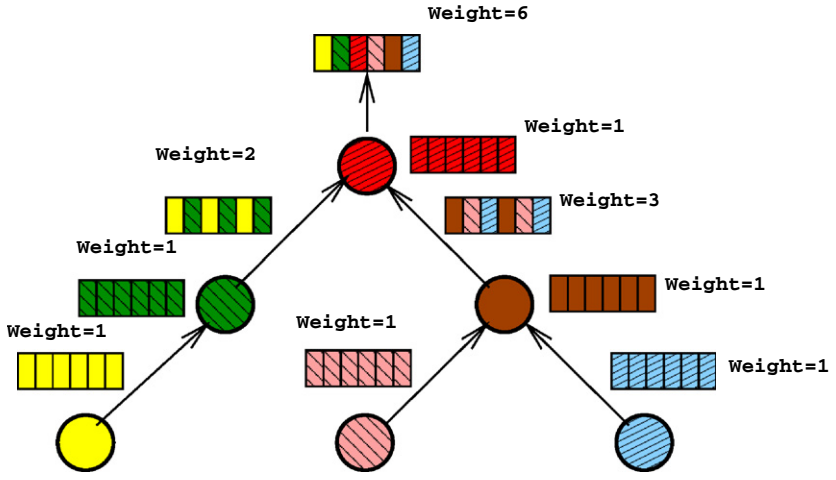

Fig. 1. An example aggregation tree showing the nodes, samples and the weights for each sample. Each node gathers samples from its own data and children's samples and creates a sample of size s. The sampling process is well distributed on each node, since each node maintains a sample of size s.

the final sample, either the bottom nodes should send more samples (vastly increasing the communication), or the samples should be weighted and the sampling algorithm has to be designed accordingly. This is the main reason existing centralized algorithms (EASE, Biased-L2) do not work on sensor networks. To overcome this difficulty in our algorithm, we introduce weights for each sample, which are simply the representations of how many other nodes this sample stands for. Deterministically sampling the gathered data using the weights, we guarantee that each node's data has the same chance to belong to the final sample, independent from its provenance in the network.

3.4. Deterministic weighted sampling (DWS)

We discussed the motivation for weighted sampling in distributed environments in the previous section. In this section, we present the main weighted sampling algorithm. DWS (pseudo-code given in Fig. 2) handles weighted sampling at a node x, where the data comes locally from D_x with a weight of 1, and otherwise from the samples in its children y, each with a weight W_y. Thus $W_x = 1 + \sum_y W_y$, and these weights can be accumulated in the network at a very low cost. Further, the weights are normalized by $\alpha_x W_x$ as in lines 1–3, and we abuse the notation slightly to use $w_j \leftarrow w_y$ for each record j coming from source y.

```
DWS(D, W, x, α_x)
 1: w_x ← 1/(α_x · W_x)
 2: for each child y do
 3:     w_y ← W_y/(α_x · W_x)
 4: for each item i do
 5:     n_i ← r_i ← 0
 6: for each record j in D_x ∪ (∪_y D_y) do
 7:     sum_n ← 0, sum_r ← 0
 8:     for each item i in j do
 9:         sum_n ← sum_n + n_i; sum_r ← sum_r + r_i
10:         n_i ← n_i + w_j
11:     R ← sum_r − α_x · sum_n
12:     K ← 2 · α_x · w_j − (2 · R)/size(j)
13:     if K > 1 then
14:         Insert j into the sample S_x
15:         for each item i in j do
16:             r_i ← r_i + 1
17: return (S_x, W_x)
```

Fig. 2. The deterministic weighted sampling algorithm

Based on the RMS distance function in Eq. (1), but allowing weighted processing in addition, the algorithm uses the following penalty function:

for each item i:

$$Q_i = (r_i - \alpha n_i)^2,$$

where r_i and n_i are the weighted counts of item i in the sample and the data set, respectively. The total penalty is $Q = \sum_{i \in I} Q_i$. For each $i \in j$, accepting a record j increases r_i by 1 and n_i by w_j, while rejecting a record only increases n_i by w_j, where w_j is the weight of the record j in data set. During sampling, each record is added to the sample with a weight of 1. After sampling finishes the record weights are updated to reflect the total weight of that sample. Since the total weight of the sample is the number of other nodes it represents, the weight is simply the total number of contributing nodes in the subtree up to the node in question, which is easily accumulated at each sampling stage.

In order to accept a record into the sample, the penalty value should decrease, namely,

$$[(r_i + 1) - \alpha(n_i + w_j)]^2 - [r_i - \alpha(n_i + w_j)]^2 < 0.$$

Refactoring the following per item condition, we obtain:

$$1 + 2(r_i - \alpha n_i - \alpha w_j) < 0.$$

Summing over all items in j, we get

$$\text{size}(j) + 2\sum_{i \in j}(r_i - \alpha n_i - \alpha w_j) < 0.$$

After further manipulation, the condition becomes

$$\text{size}(j) + 2\sum_{i \in j}(r_i - \alpha n_i) - 2\alpha w_j \cdot \text{size}(j) < 0.$$

The acceptance rule then becomes

$$2\alpha w_j - \frac{2\sum_{i \in j}(r_i - \alpha n_i)}{\text{size}(j)} > 1$$

which is the acceptance rule used in Fig. 2 (lines 12–13).

Our method works by extracting from the sensor data a set of multi-dimensional boolean attributes. Categorical attributes are easily reduced to this

setting by introducing a boolean characteristic attribute for each discrete category; numerical attributes can be reduced to it as well by associating a boolean attribute to a range in some shared histogram. Each boolean attribute is considered as a subset of the sensors (those for which there is a match). It strives to select a sample that simultaneously minimizes the frequency error over all the attributes.

As the communication costs are the dominating factor in sensor networks, we present the performance of the algorithm in number of messages. Assuming each sample value fits in a single message, the sample size of each node is s, and the total number of nodes is k, our algorithm performs a full sampling with $O(ks)$ messages. The memory requirement of the algorithm is $O(m + sT_{\text{avg}})$ per node, based on storing the counts for each item (m such items), weights for each record and the raw sample. Practically speaking, if we assume we are using a data set with 5 tuples per record (similar to our real world data set described further in Section 4.1), we need to use 6 integers (24 bytes) to store a record with its weight. Typically, keeping a sample of 1000 records requires 24 Kbytes. The space required for the item counts is flexible and independent from the sample size. We can use a higher granularity for item counts to increase the accuracy of our sample, or if application permits, we can decrease the granularity to save memory for additional sample values. For example, if we use 100 equi-sized buckets per tuple to represent item counts, we need 2 Kbytes to store integer counts for the items. Since a typical sensor node (ex. Berkeley Motes) currently has 128 Kbytes of memory, we assume these are reasonable memory requirements.

4. Experiments

In this section, we now evaluate our DWS algorithm both on real and synthetic data, and compare with the DIR sampling algorithm [23] based on sample accuracy and energy usage. We demonstrate our work using the wireless sensor network simulator Shawn [30]. Shawn is a discrete event simulator that allows us to test our algorithms on more nodes than the other well-known simulators.

We introduce the real world climate data set in Section 4.1. In Section 4.2 we give the simulation results of our deterministic algorithm (DWS), and compare the results to other algorithms such as naive random sampling and duplicate insensitive random (DIR) sampling. We show that our algorithm has many advantages such as better sample quality and less communication rate compared to other methods. Finally, Section 4.3 demonstrates the performance of our algorithm in answering typical SQL queries such as AVG, COUNT, SUM, MAX, etc. over climate data set.

All energy usage calculations in the experiments are based on total number of sent and received messages. Using the weights from [29], the received messages are weighted with 1.2, the sent messages are weighted with 1.7 and energy usage for a node is the total. Each tree building message is calculated as a single message. When exchanging samples between nodes, each record is assumed to fit in exactly one message, also the weights of the whole sample are assumed to fit in one message. These are reasonable assumptions, since there is only limited number of sensors on each node, so the nodes can transfer all readings in one single message.

4.1. Climate data set

In our experiments, we used real world climate data from US National Climatic Data Center.[1] The data set is the daily readings of precipitation, snow fall, snow amount, maximum and minimum temperature values throughout the continental US, covering 1064 stations for nearly the last 100 years. Since there are gaps in the station readings, we selected the maximum possible stations covering the exact same days. We gathered 227 stations having 4153 daily readings per station. The time period of the readings is presented in Fig. 4(left). The black marked regions show the available days while the white regions show the gaps between daily readings. The real locations of the 227 stations on the US map[2] is also presented in Fig. 4 (right).

For our real data experiments, we simulated the sensors on the US map (scaled) by associating each sensor with a station, and positioning 227 sensor nodes on a 60×25 region, according to real station positions (latitude and longitude) on the map. The sensor node corresponding to the station in Central Park, NY is selected as the sink. To preserve the time and location correlation of data, we further fed

[1] http://www.ncdc.noaa.gov/oa/climate/research/ushcn/daily.html

[2] Generated using tools from http://www.planiglobe.com

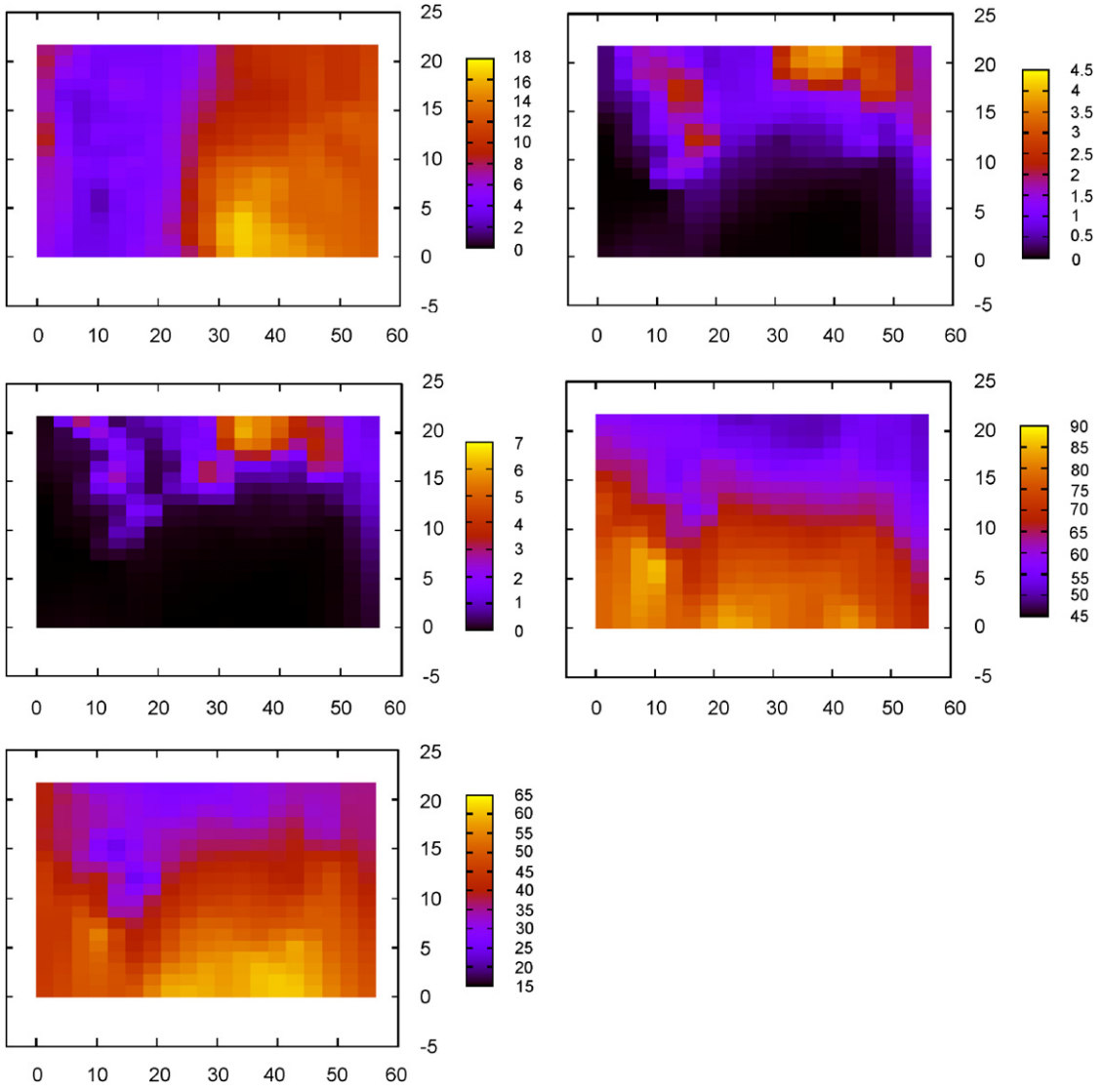

Fig. 3. Average values of precipitation (top left), snow fall (top right), snow amount (center left), maximum temperature (center right), and minimum temperature (bottom left).

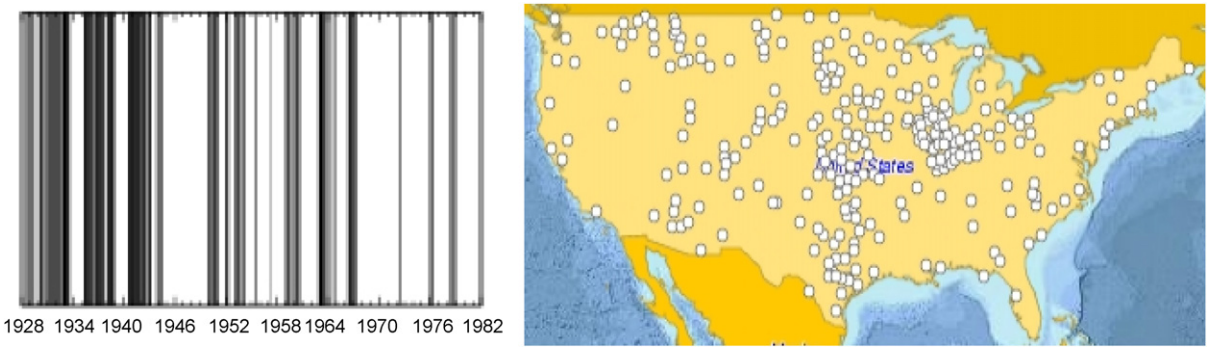

Fig. 4. Time period for weather data used in the simulations (left). Simulated US area (60 × 25), and 227 sensor locations (right).

each sensor with the exact data collected from the associated station, in original order. This way, each reading of a sensor corresponds to a daily data of the associated station. The average values of the 5 tuple data are plotted in Fig. 3.

Analyzing the climate data, we also realized that there exist many inconsistencies within the data, because of measurement equipment failures in early days, and metric conversion errors. We left these inconsistencies as is, since they present additional challenge for us in the data set, and these kind of perturbations in data are quite likely in real world sensor deployments.

4.2. Distributed data reduction

In this section, we give the simulation results of our Distributed Weighted Sampling algorithm on sensor networks. The evaluation is based on two categories, the sample quality and the energy usage. In order to demonstrate our work, we also include the naive random sampling algorithm and a more advanced, DIR sampling algorithm from Nath et al. [23].

Two different data sets are used in the simulations: the climate data set we introduced in Section 4.1, as the real world data set and the synthetic data set (T10I6D1500K), which is an association rule data set, created using the IBM data generator [31]. The synthetic data set includes $1,500,000$ records, each having different number of items. The average number of items in each record is 10, the average number of itemsets is 6, and the total number of unique items is 1000. Items represent different sensor readings, and a record represents a total reading from a sensor, with various items. Items represent discretized readings from sensors; for instance, if we assume we have 10 sensors on a single node, and each sensor reading is discretized into 100 buckets (such as temperature), a record having items 30, 140, 550 means the readings for the first sensor is 30, second one is 40, fifth one is 50 and there are no readings present for the rest of the seven sensors.

For the simulations with the climate data, we used 227 nodes, each having a constant radio range of 6 to ensure connectivity. The sink is placed on a predefined location as discussed in Section 4.1. For synthetic data set simulations, we again used 227 nodes, randomly deployed in a 60×25 area, sink on the center.

Initially, for each algorithm, a data aggregation tree is created using the same algorithm as in [19]. In naive random sampling, each node samples s records and forwards the samples up the tree without any further data reduction. At the end the sink has sk records from a network of k nodes, which it further samples to generate a sample of size s. The details of DIR algorithm are already discussed in Section 2, and our DWS algorithm in Section 3.

The simulation results show the averages of running DWS algorithm on synthetic data set and random algorithms (naive and DIR) on both data sets 100 times. Since climate data set is a static data set, and DWS is a deterministic algorithm, DWS is ran once on this data set.

Fig. 5 (left) and Fig. 6 (left) show the RMS error values of the final sample generated by the algorithms for four different sample sizes 1000, 2000, 3000 and 4000 on both data sets. From these figures we can see that our DWS algorithm generates up to a factor of 4 times better quality samples than the other two algorithms, for both data sets.

Fig. 5 (center) and Fig. 6 (center) show the total energy used in the network while generating the samples. The energy usage is calculated based on the

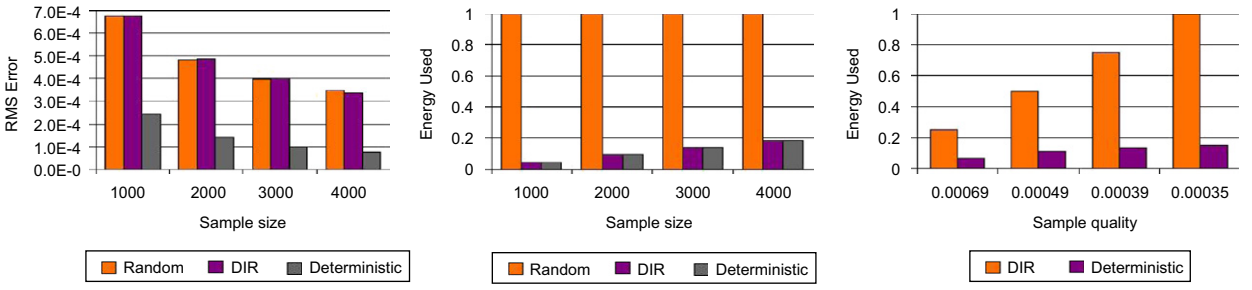

Fig. 5. Results for real world climate data set. (Left) RMS results vs. sample size, (center) energy usage vs. sample size, and (right) energy usage vs. sample quality.

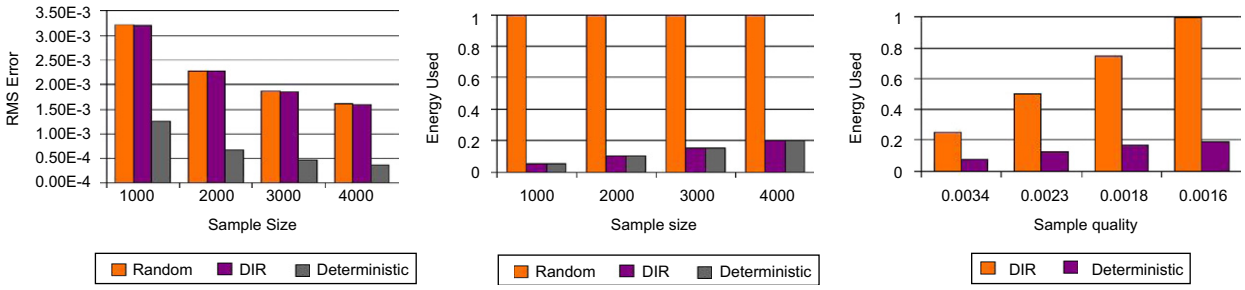

Fig. 6. Results for synthetic data set. (Left) RMS results vs. sample size, (center) energy usage vs. sample size, and (right) energy usage vs. sample quality.

total number of messages sent and received as described in Section 4. The energy usage results are scaled for a clear presentation. As expected, the energy usage of DIR and DWS are the same, and up to a factor of 20 times less than that of the naive random sampling.

To clearly compare DIR and DWS algorithms, in Fig. 5 (right) and Fig. 6 (right), we generate two samples having the same quality based on RMS error values and compare the energy used by both algorithms. Here, DIR algorithm still uses samples of sizes 1000, 2000, 3000 and 4000. The sample sizes of DWS algorithm are 268, 448, 532, and 600 for climate data set, and 314, 505, 685, and 766 for synthetic data set. For both data sets, DIR algorithm uses up to a factor of 6 times more energy than DWS algorithm. From the figures it is clear that, we can have the same quality sample by using considerably less energy if we use our DWS algorithm.

4.3. Example SQL queries

In this section, we evaluate answering SQL queries using the sample instead of the data set after the sample is extracted and stored in a DBMS outside of the network. We presented sample quality results of our algorithm in the previous section. Those results were based on RMS distance between the data set and the sample. Now we compare the two algorithms using AVG, MAX, SUM and COUNT queries. To achieve this goal, we use the real-world climate data set as our data set of choice. As we keep the original order of the records in this data set, we created a single sample of size 4000 (sampling rate of 0.0036) using our deterministic algorithm, and 10 samples of size 4000 using the DIR sampling algorithm (as this is a randomized algorithm). Our deterministic algorithm on this static data always creates the same sample, so it is not necessary to generate multiple samples. On the other hand, the DIR algorithm is run multiple times to observe the average behavior of the algorithm, since the samples generated are different for each run of the algorithm. When presenting the results for the DIR algorithm, we present the min, max, mean and standard deviation of the errors over the 10 samples.

Samples are created once and all four queries are answered using these samples. The evaluation metric is the error rate between the query results of the data set and the sample, calculated as

$$\text{Error Rate} = \left|\frac{\text{Result}_{\text{data set}} - \text{Result}_{\text{sample}}}{\text{Result}_{\text{data set}}}\right|.$$

In Fig. 7, the horizontal lines are the DWS error rates, which is a single value per sample. The vertical lines are the min–max values for DIR error rate, and the boxes show the mean and the standard deviation values of the DIR error. Table 1 presents in detail the comparison of the DWS and DIR algorithms. DWS errors are compared with the mean, min and max errors of the DIR samples, and the results are presented as a ratio of $\text{DIR}_{\text{error}}/\text{DWS}_{\text{error}}$ for easy comparison.

Query 1:

```
SELECT AVG(prcp), AVG(snow),
AVG(snwd), AVG(tmax), AVG(tmin)
FROM data set
```

This is a query to test the effects of sampling algorithms on AVG queries. The query selects the average values for all 5 tuples on both data set and each sample, and Fig. 7 (top left) and Table 1 show the error rates of both algorithms. Looking at the average error rates, except snow fall (snow), DWS algorithm outperforms DIR, sometimes by a factor of 24 times. Additionally, DWS error rate is much

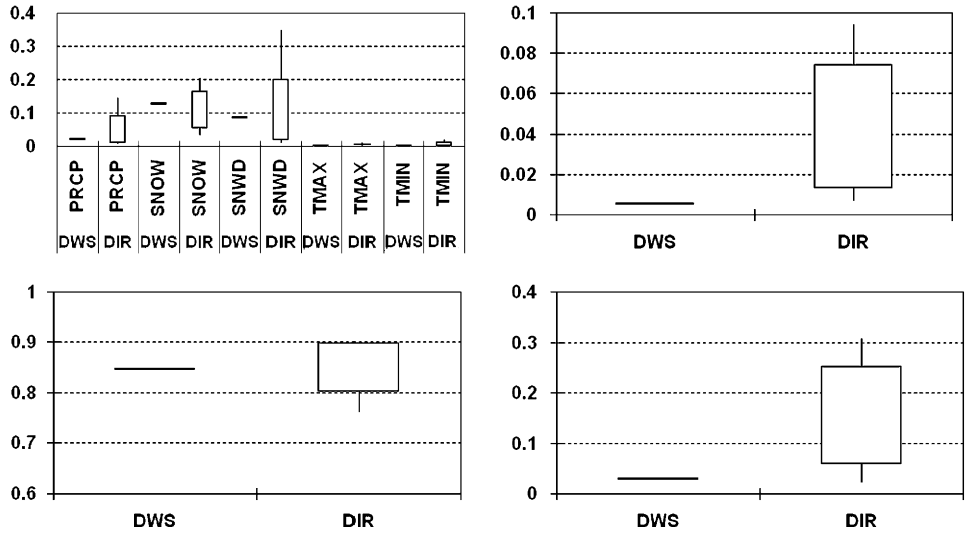

Fig. 7. SQL query errors for deterministic sample and DIR samples. Results for Query-1 (top left), Query-2 (top right), Query-3 (bottom left) and Query-4 (bottom right).

Table 1
Ratio of the mean, min and max error values of the DIR samples to the error values of DWS sample for each query

DIR/DWS	Query-1					Query-2	Query-3	Query-4
	PRCP	SNOW	SNWD	TMAX	TMIN			
MEAN	2.50	0.86	1.31	5.33	24.41	8.39	1.00	5.47
MIN	0.33	0.26	0.12	0.21	3.56	1.35	0.90	0.77
MAX	7.20	1.60	4.13	9.74	59.26	18.17	1.06	10.85

smaller than the worst case DIR samples (sometimes by a factor of 59 times), and close to the minimum error range of random samples. We would like to state here that it is unlikely for a single random sample to accurately answer all queries, the minimum error values for DIR are coming from different random samples; however, we use only one DWS sample and being close to the minimum bounds of DIR samples shows how accurate our deterministic sample is.

Query 2:

```
SELECT COUNT(*)
FROM data set
WHERE snow = 0 AND snwd > 0
```

The second query finds the number of days when the snow fall amount is zero but there exist snow on the ground. Our focus is to test COUNT type queries. We are also testing if the correlation information between snow fall (snow) and snow amount on ground (snwd) are preserved accurately in the samples. Fig. 7 (top right) and Table 1 show the results for this query. As we claimed, the deterministic algorithm is more accurate to represent the correlation information in multi-dimensional count data. The deterministic sample is a factor of 8 times better than the average, 1.3 times better than the best and 18 times better than the worst random sample. Deterministic sample is the clear choice in this query as it outperforms random samples in all aspects.

Query 3:

```
SELECT MAX(prcp)
FROM data set
WHERE tmax > 80
```

This is an outlier query testing the MAX value and also the correlation information between precipitation (prcp) and maximum temperature (tmax). The result is presented in Fig. 7 (bottom left) and Table 1. Both algorithms are quite inaccurate finding the MAX value. Note that this result is typical, samples in general are not good for handling outliers. Detecting and handling outliers

is another research topic [3,4], and requires more specific data structures and techniques. We also would like to highlight that each of these specific techniques solve a specific kind of outlier problem, and there is no "one size fits all solution". Therefore, all these techniques can be used additional to the sampling approach to generate a more complete data reduction solution. For this reasons, we only would like to demonstrate the shortcomings of sampling in general, and focus on finding samples for more general use.

Query 4:

```
SELECT SUM(snow)
FROM data set
WHERE tmin < 0
```

The last one is to test SUM type of queries. Selects the total snow fall amount for days having minimum temperature less than 0°. The results are in Fig. 7 (bottom right) and Table 1. We can clearly see that, for this query, the deterministic sample is slightly worse than the best random sample, and a factor of 5 times better than the average and also a factor of 10 times better than the worst random sample. In this query also, deterministic sample is much more accurate in keeping the correlation information between snow fall and minimum temperature.

We tested our algorithms on real world data set, using typical SQL queries. As expected the sampling algorithms give accurate results for SUM, COUNT, AVG type of queries, and quite inaccurate results when it comes to outlier queries such as MAX (see Fig. 7 and Table 1). We also observe that deterministic algorithm is much more accurate in keeping correlations between items. The accuracy of the aggregation algorithm relates to the total energy usage of the network (communication to extract the sample), such as a smaller deterministic sample is more preferable to a bigger random sample, when both have the same accuracy. The results in Fig. 7 and Table 1 also demonstrate that once a sample is extracted from the sensor network, it can be used to answer different types of queries (except outliers of course), with reasonable accuracy.

5. Concluding remarks and future work

We have presented a novel deterministic weighted sampling algorithm as a new aggregation method for network of wireless sensors. DWS is simple enough to not consume too many resources locally, and we validate through experiments that the sample it provides is vastly superior to other distributed sampling methods exist for sensor networks. Our algorithm is designed to work on arbitrary network topologies, by introducing weights for samples and dynamically updating these weights throughout the sampling. DWS by design effectively distributes the aggregation work over all the nodes by enabling each node to generate a fixed sized sample and prevents any node from being a bottleneck.

One criticism of our approach is that loss of connection in the aggregation tree structure induced by link or node failure can have drastic effects on the representativity of our sample, since an entire subtree may no longer be contributing to the sample. This can be handled by allowing a multi-path aggregation structure [23,24]. In the context of aggregation, however, one must then address the problem posed by the duplication of the data along these multi-paths. The duplicate-insensitive solutions provided by Nath et al. [23] and Considine et al. [22] do not extend to our deterministic algorithm, and we leave it as a matter for future research to handle robust connectivity with our deterministic sampling algorithm.

Although our original design requires input from every sensor node, this can easily be remedied by incorporating other sparse sampling approaches. We note that our approach only concerns how the samples are aggregated along the network. For instance, the contributing nodes can be selected by another sampling approach such as the approximate uniform random sampling of Bash et al. [32]. Likewise, the temporal frequency at which measurements are updated at a node can be regulated by an adaptive process such as proposed by Jain and Chang [33]. In any case, known compression methods can be applied to the in-network sample aggregation to further minimize communication costs [27,28,34].

References

[1] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, Wireless sensor networks for habitat monitoring, in: Proceedings of 1st ACM Workshop on Sensor Networks and Applications (WSNA'02), 2002, pp. 88–97.

[2] K. Levon, Detection of biological warfare agents, in: Proceedings BCC Nanotech and Biotech Convergence 2003 Conference, Stamford, CT, 2003, pp. 169–186.

[3] D. Barbará, W. DuMouchel, C. Faloutsos, P.J. Haas, J.M. Hellerstein, Y.E. Ioannidis, H.V. Jagadish, T. Johnson,

R. Ng, V. Poosala, K.A. Ross, K.C. Sevcik, The New Jersey data reduction report, IEEE Data Eng. Bull. 20 (4) (1997) 3–45.

[4] P.B. Gibbons, S. Acharya, Y. Bartal, Y. Matias, S. Muthukrishnan, V. Poosala, S. Ramaswamy, T. Suel, Aqua: system and techniques for approximate query answering, Technical Report, Bell Labs, 1998.

[5] H. Akcan, A. Astashyn, H. Brönnimann, L. Bukhman, Sampling multi-dimensional data, Technical Report TR-CIS-2006-01, CIS Department, Polytechnic University, February 2006.

[6] H. Brönnimann, B. Chen, M. Dash, P.J. Haas, P. Scheuermann, Efficient data reduction with EASE, in: Proceedings of ACM KDD'03, 2003, pp. 59–68.

[7] B. Chen, P.J. Haas, P. Scheuermann, A new two-phase sampling based algorithm for discovering association rules, in: Proceedings of ACM KDD'02, 2002, pp. 462–468.

[8] H. Toivonen, Sampling large databases for association rules, in: Proceedings of VLDB'96, 1996, pp. 134–145.

[9] M.J. Zaki, S. Parthasarathy, W. Lin, M. Ogihara, Evaluation of sampling for data mining of association rules, Technical Report 617, University of Rochester, Rochester, NY, 1996.

[10] G. Manku, R. Motwani, Approximate frequency counts over data streams, in: Proceedings of VLDB'02, 2002, pp. 346–357.

[11] H. Akcan, H. Brönnimann, Deterministic data reduction in sensor networks, in: Proceedings of the Third IEEE International Conference on Mobile Ad-hoc and Sensor Systems, 2006, pp. 530–533.

[12] H. Brönnimann, B. Chen, M. Dash, P.J. Haas, Y. Qiao, P. Scheuermann, Efficient data-reduction methods for on-line association rule discovery, in: Chapter 4 of Selected Papers from the NSF Workshop on Next-Generation Data Mining (NGDM'02), MIT Press, Cambridge, MA, 2004, pp. 190–208.

[13] D.J. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, S.B. Zdonik, The design of the Borealis stream processing engine, in: Proceedings of CIDR, 2005, pp. 277–289.

[14] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, R. Motwani, I. Nishizawa, U. Srivastava, D. Thomas, R. Varma, J. Widom, STREAM: the stanford stream data manager, Trans. IEEE Data Eng. Bull. 26 (1) (2003) 19–26.

[15] S. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, The design of an acquisitional query processor for sensor networks, in: Proceedings of SIGMOD'03, 2003, pp. 491–502.

[16] Y. Yao, J. Gehrke, The cougar approach to in-network query processing in sensor networks, SIGMOD Record 31 (3) (2002) 9–18.

[17] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy efficient communication protocol for wireless micro-sensor networks, in: Proceedings of 33rd Hawaii International Conference on System Science, 2000, pp.8020.

[18] S. Lindsey, C. Raghavendra, K.M. Sivalingam, Data gathering algorithms in sensor networks using energy metrics, IEEE Trans. Parallel Distributed Systems 13 (2002) 924–935.

[19] S. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, TAG: a tiny aggregation service for ad hoc sensor networks, in: Proceedings of USENIX (OSDI), 2002, pp. 131–146.

[20] M.A. Sharaf, J. Beaver, A. Labrinidis, P.K. Chrysanthis, Balancing energy efficiency and quality of aggregate data in sensor networks, VLDB J. 13 (4) (2004) 384–403.

[21] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, An application specific protocol architecture for wireless micro-sensor networks, IEEE Trans. Wireless Comm. 1 (4) (2002) 1.

[22] J. Considine, F. Li, G. Kollios, J. Byers, Approximate aggregation techniques for sensor databases, in: Proceedings of IEEE ICDE'04, 2004, pp.449.

[23] S. Nath, P.B. Gibbons, S. Seshan, Z.R. Anderson, Synopsis diffusion for robust aggregation in sensor networks, in: Proceedings of SenSys'04, Baltimore, MD, 2004, pp. 250–262.

[24] A. Manjhi, S. Nath, P.B. Gibbons, Tributaries and deltas: efficient and robust aggregation in sensor network streams, in: Proceedings of SIGMOD'05, Baltimore, MD, USA, 2005, pp. 287–298.

[25] N. Shrivastava, C. Buragohain, D. Agrawal, Medians and beyond: new aggregation techniques for sensor networks, in: Proceedings of SenSys'04, 2004, pp. 239–249.

[26] M.B. Greenwald, S. Khanna, Power-conserving computation of order-statistics over sensor networks, in: Proceedings of PODS'04, 2004, pp. 275–285.

[27] I. Lazaridis, S. Mehrotra, Capturing sensor-generated time series with quality guarantees, in: Proceedings of ICDE, 2003, pp.429.

[28] A. Deligiannakis, Y. Kotidis, N. Roussopoulos, Compressing historical information in sensor networks, in: Proceedings of SIGMOD, 2004, pp. 527–538.

[29] B.J. Chen, K. Jamieson, H. Balakrishnan, R. Morris, Span: an energy efficient coordination algorithm for topology maintenance in ad-hoc wireless networks, Wireless Networks 8 (5) (2002) 481–494.

[30] A. Kröller, D. Pfisterer, C. Buschmann, S.P. Fekete, S. Fischer, Shawn: a new approach to simulating wireless sensor networks, in: Proceedings of Design, Analysis, and Simulation of Distributed Systems (DASD05), 2005, pp. 117–124.

[31] Intelligent Information Systems, Synthetic data generation code for associations and sequential patterns, Research group at the IBM Almaden Research Center.

[32] B.A. Bash, J.W. Byers, J. Considine, Approximately uniform random sampling in sensor networks, in: Proceedings of First Workshop on Data Management in Sensor Network (DMSN'04), Toronto, Canada, 2004, pp. 32–39.

[33] A. Jain, E.Y. Chang, Adaptive sampling for sensor networks, in: Proceedings of DMSN'04, 2004, pp. 10–16.

[34] A. Deligiannakis, Y. Kotidis, Data reduction techniques in sensor networks, IEEE Data Eng. Bull. 28 (1) (March 2005) 19–25.