Research Directions in Sensor Data Streams: Solutions and Challenges

Eiman Elnahrawy
Department of Computer Science
Rutgers University
eiman@paul.rutgers.edu

Abstract

A typical framework of sensor streams is data obtained from wireless networks of sensors, embedded in a physical space, continuously communicating a stream of data to a database. These wireless networks typically consist of large number of low-power and limited-bandwidth devices. They are primarily used for monitoring of several physical phenomenon such as, contamination, climate, building structure, etc., potentially in remote harsh environments. Research in sensor streaming has been generally focused on ultimate utilization of such devices given their limited resources and unattended deployment. This paper surveys current research directions in sensor data streams. In particular, it emphasizes existing work on storage and gathering of sensor data, architectures for querying sensor streams, and handling of erroneous sensors. It also highlights some open problems and discusses research paths to pursue in this exciting research area.

1 Introduction

Wireless sensor networks typically consist of few thousands of sensors, embedded in physical spaces, continuously collecting and communicating their data stream to the database. The database usually resides at a "powerful" device called a base-station. Such sensors are low-power, low-bit rate devices, and usually sampled at low rate, i.e., few times per second or less, depending on various factors such as the application, type of sensors, etc. They are currently used in several real life applications. Specifically, for monitoring several physical phenomena such as climate, e.g., light, temperature, wind, etc., concentration of contaminants, building structure and response to earthquakes, etc., especially in remote hard to administer environments. Due to their low cost (can be as cheap as 10 cents), these devices are expected to become pervasive, and consequently, to be a major source of information for databases. In particular, in the near future, every object will afford to have a sensor on it. These sensors will operate as black-boxes that record diagnostic data, performance data, history of the object, etc., e.g., in vehicles, cell phones, bridges and intersections, computers, inventory, and so on. Therefore, the future of sensor streams lies in reasoning about such data and solving any existing problems that prevent their wide deployment.

The major focus of the current research on sensor streams, among the database community, is data gathering techniques using network primitives, e.g., [12, 7, 20, 19]. This research takes into consideration the severe resource constraints of sensor networks, especially energy constraint, and their unattended deployment potentially in harsh environments. Our ongoing work, on the other hand, is primarily focused on online data cleaning techniques such as cleaning and querying of noisy data, discovering outliers, and handling incomplete data due to missing values, e.g., [5]. Our motivation is that such problems generally limit the deployment of sensors in the real world as "reliable" sources of information, specifically for decision making. This paper surveys current research directions in sensor data streams. It discusses current work on storage of sensor data streams, aggregation of sensor streams using in-network distributed techniques, frameworks for querying sensor streams, and

finally, our recent work on handling of erroneous sensors. It also emphasizes some challenges and open problems in this area that need further investigation and highlights exciting research paths to pursue while dealing with sensor data.

The rest of this paper is organized as follows. In Section 2, we give an overview of wireless sensor networks, their limitations and capabilities, and compare sensor data streams to traditional streams. We present existing techniques for storage of sensor data in Section 3. Section 4 surveys existing research on in-network aggregation of sensor streams. We present Fjord, an architecture for queries over streaming sensor data in Section 5. We then discuss our ongoing work on handling imprecision in sensor networks in Section 6. Finally, we conclude this paper and discuss our view of the future of sensor streams in Section 7.

2 Background

In this section, we discuss limitations of wireless sensor networks. We also highlights some of the major problems of sensor data obtained from such networks. We then emphasize the characteristics of sensors considered in this survey, and distinguish between sensor streaming and traditional data streaming.

2.1 Limitations and Problems

Unfortunately, sensors of wireless sensornets have serious resource constraints [7, 12, 22]. In particular, they have limited battery life, which, if abused, may cause the sensors to live for only few days as opposed to few months. In addition, they have constrained communication bandwidth (1-100 Kbps), and limited storage and processing capabilities. For example, a typical sensor has 4MHz processor, 8KB programming memory, 512B data memory, 10 Kbps Bandwidth only [8]! These energy, bandwidth, storage, and processing limitations enforce special data handling algorithms and architectures for sensor data streams, that explicitly incorporate these resource constraints. Potti *et al* have shown that communication cost in sensor networks is orders of magnitude higher than their processing cost [17], and therefore, current approaches for handling of sensor data usually strive to minimize energy, spent by the sensors, by preprocessing this data. This preprocessing aims at minimizing the size of the data before communicating it to the base-station (database), and consequently, reduce communication energy. We will later discuss several such approaches throughout this paper.

Another serious problem of sensor data streams is incomplete data. In particular, existence of missing values among data obtained from wireless sensor networks. There are many factors that contribute to this problem such as packet loss and topology changes. The former arise in wireless sensornets due to poor links and communication failures, fading of signal strength, packet collision between multiple transmitters, and constant or sporadic interferences [22]. Zhao *et al.* have shown the severeness of this problem experimentally. Specifically, they found that more than 10% of the network links suffer average loss rate $> 50\%$, and that packet loss of most links fluctuates over the time with estimated variance $9\% - 17\%$ [22]. Nevertheless, the topology of a sensornet is usually continuously changing due to node failures and node movements, e.g., based on our experimental deployments, on the average, $5 - 10\%$ of the nodes may be assumed failed at each sampling attempt. Most of the existing research on missing values is either focused on providing low-level networking solution such as [22], or customized solutions that work for specific applications such as [13, 12], which is considered a limited solution. In both cases, the problem persists, although less severely. Our ongoing research, on the other hand, is focused on a general purpose solution for this problem.

Finally, another prevalent problem in sensor data is imprecision, either due to lag of database update or due to noisy readings. In the former case, the massiveness of readings and the limited energy and wireless bandwidth may not allow for continuous and instantaneous updates, and therefore, the database state may lag the state of the real world. This problem has been addressed recently in [4]. The later case, however, is due to inaccuracies of measurements which is the primary source of data. The sources of inaccuracies include, but are not limited to: (a) noise from external sources, (b) inaccuracies in the measurement technique, (c) calibration errors, and (d) imprecision in computing a derived value from the underlying measurements. The cost of imprecise data can be very significant if they result in an immediate decision making or actuator activation. Imprecision due to noisy

readings is considered an important cause of uncertainty in sensor databases, and hence, we are addressing this problem in our ongoing research [5].

2.2 Capabilities

It is worth distinguishing between sensors used in wireless networks, also called "smart dust", which are low-power and have limited resources, and larger more powerful sensors, e.g., sensors considered in [15], or passive sensors, e.g., [1]. The former type of sensors, i.e, smart dust, represent the current generation of sensors, and they are the main focus of this survey. Powerful sensors, on the other hand, are usually far more expensive than smart dust, and therefore, they are of limited usage. Our goal, however, is to have very cheap sensors, scattered everywhere, collecting data about various real life phenomena continuously. Some of the algorithms and techniques, that will de discussed in this paper, are useful for such powerful sensors as well. Specifically, discussion of Fjords in Section 2, and noisy sensors in Section 6. To the best of our knowledge, there is no existing research on modeling of passive sensors or designing of frameworks for querying such sensors. We think that this is an interesting research path to pursue.

Capabilities of sensors in wireless networks varies slightly based on the type of sensors, the nature of queries, and consequently, the applications. In general, these sensors can perform simple tasks such as forwarding their raw readings to a base-station or nearby sensors, performing simple aggregations of their own readings, or performing simple partial processing. Their power and processing resources enable limited storage of some of their own readings or readings of neighboring sensors locally. I.e., such sensors are capable of constructing and maintaining a locally consistent view of its neighborhood [11].

2.3 Sensor Streaming versus Traditional Streaming

We have discussed several limitations of wireless sensor networks and problems of sensor data, obtained from such networks. In this section, we argue that existing traditional streaming techniques are not directly applicable to sensor streaming, and we highlight some of the major differences between the two areas. The first distinction is that data of sensor streams are only samples of the entire population. The sampling rate varies from application to another, e.g., temperature and light data are usually sampled several times per second, while contamination is sampled at much lower frequency [14]. On the other hand, the entire population is usually available in traditional streaming, e.g., data of web logs, network streams, stock market, etc. Second, sensor data is usually imprecise and noisy, while traditional streaming data is exact and error-free [5]. Third, existing sensor streams is typically of moderate size as compared to overwhelming storage and processing of huge data in traditional streams. This distinction, however, is tied to the current applications of sensor streams which involve data of moderate size, i.e., few thousands of tuples at a specific time instance. Our conjecture is that sensors will become very pervasive in the future, and consequently, sensor streams will be very huge. We will discuss a scalable architecture for sensor streams in Section 5, and we will further discuss our predictions of the future of sensor streams in Section 7. Fourth, due to the fact that sensors die when they run out of power, data acquisition in sensor networks is expensive, while data of traditional streaming is considered free with no acquisition cost. Finally, sensor data has continuous domains (i.e., continuous attributes). As such, some of questions posed in traditional streaming such as frequent items, distinct values, etc., are clearly not meaningful for such domains. Current applications of wireless sensor networks, on the other hand, usually involve "simple" aggregates. We believe that this will continue to be the case even when sensors become pervasive.

3 In-network Storage

In this section we discuss recent research on storage of sensor data in the network. Generally, we are interested in systems that extract data from sensor fields and enable users to observe, analyze, and query this data. These systems should be (1) energy-efficient, (2) scalable in the size of the network, and (3) self-organizing and robust against node failures and topology changes at the same time. In order to minimize energy consumption,

communication cost needs to be optimized, when storing sensor readings and when querying. Specifically, systems should benefit from general characteristics of sensor data, such as their spatio-temporal nature, to reduce the size of data communicated to the database[1]. Scalability can be achieved by adopting hybrid systems that combines distributed and hierarchical structures. Finally, robustness is generally achieved by data redundancy and low-level networking.

Ganesan *et al.* argue that existing systems do not satisfy these design goals especially energy-efficiency and scalability, and therefore, they introduce an alternative architecture for handling of sensor data [7]. For example, they argue that the objective of hierarchical web caches is to lower load and latency in network traffic by strategically placing frequently accessed web pages. However, hierarchical web caches are not designed for resource constrained systems as in wireless sensor networks. They also do not utilize space and time correlations between their data (i.e., web pages) which is common among sensor readings. Another argument is that Geographic Information Systems (GIS), which handles spatio-temporal data, have "centralized" processing and their design goal is to reduce data search cost, irrespective of the energy cost. They also argue that although the spatio-temporal characteristic of data in media streaming systems, on the internet, is similar to sensor data, they are based on space first time next compression. The objective of the space-wise compression is to compress each data frame while the objective of the time-wise compression is to compress the value of each point in the data over successive frames. A time first space next compression, on the other hand, is more resource-efficient in sensor networks since time compression is performed locally at the sensor node, while space compression requires communication between different nodes, and consequently, energy. Finally, they argue that cost metric in lossy compression using wavelets should be a local metric that tradeoff compression versus communication and not compression versus reconstruction error.

Given our desired design goals, there are generally three approaches for storing data in sensor networks: external, local, and data centric storage approaches [19, 18]. In external storage, sensor data is continuously sent to a "powerful" collecting point(s). Hence, there is a cost for communication of data to the base-station, however, there is no cost of querying this data (with respect to sensor energy consumption[2]). On the other hand, data in local storage are stored locally at its original node. Consequently, there is no cost of data communication, however, there is a significant cost of querying this data since queries will be sent to every node in the network (e.g., by flooding). Finally, in data centric storage (DCS), data are stored by name, and therefore, there is a cost of communicating the data from its original node to the node where it will be stored. However, cost of querying in this case is reduced since queries are directed to the node that stores the data and not to every node in the network. As a conclusion, there is a significant tradeoff between energy consumption and data storage technique. To decide which storage technique to use in a specific deployment of sensors we need to have a prior knowledge of the network characteristics and the deployment. For example, the topology of the network and its use, the number of nodes, the sensed phenomena, and the application (and therefore, the nature of queries), are all important factors that will affect our decision.

In this section, we discuss research on the three major storage approaches, an external storage approach [9], a local storage approach [7], and a data centric storage approach (DCS) [19, 18]. The surveyed systems assume that queries are either summaries of sensor readings or detailed data sets of readings, however, there is no explicit definition of summary queries. Goel *et al.* are interested in obtaining detailed data sets from sensor networks for external storage [9]. They propose an energy-efficient paradigm for collecting detailed sensor data called "PREdiction-based MONitoring" or PREMON. PREMON utilizes spatio-temporal correlations in sensor data in order to reduce the size of data communicated from the sensors to the base-station. Specifically, PREMON reduces communication cost by predicting future sensor data, so called "prediction-model", at the powerful base-station, then sending these predictions to the sensors. The sensors are instructed to not send their

[1] Recall that processing cost is negligible compared to communication cost [17].

[2] Generally, we assume that we are only interested in energy consumed by sensor nodes, i.e., any other energy consumption has no cost.

reading to the base-station if the readings are within a predefined threshold from the prediction. Their approach is inspired by the similarities between sequential frames of detailed sensor readings and MPEG movies. Ganesan *et al.*, on the other hand, are interested in systems that enable multi-resolution queries on sensor data, i.e., summaries as well as detailed data sets [7]. Specifically, they propose a unified view of sensor data handling systems that incorporates local long term storage, multi-resolution queries, and efficient support for spatio-temporal pattern mining. They emphasize the importance of considering spatio-temporal correlations between sensor readings when designing architectures for such systems, in cost models, and in metrics of evaluation. They experimentally show that utilizing such spatio-temporal correlations in sensor data can significantly reduce its size via compression. Hence, they provide a distributed, hierarchical, and multi-resolution long term storage that is based on hierarchical wavelet decomposition. The last contribution in this area is due to Ratnasamy *et al.* [19, 18]. They introduce a DCS-based data dissemination algorithm for storing sensor data. Their focus is on sensors deployed in harsh environments such as habitat monitoring and so on. The identity of the sensors which collected the data, in their approach, is less relevant compared to the collected data. Although this is not usually a realistic assumption, it is appropriate in some applications such as habitat monitoring and tracking of animals, where data represents animal motion while queries involve events about tracking specific animals. Therefore, sensor data is "named" and accessed via its name using a data centric technique. All data with the same general name are stored at the same sensor node, not necessarily the one that collected the data. Therefore, queries to specific data can be sent directly to the node that store the data rather than flooding the entire network with the query. In the rest of this section we discuss each approach in more details, highlight its limitations, and discuss some open problems that need to be solved.

3.1 Prediction-based Models for External Storage

PREMON aims at collecting detailed data sets from wireless sensor networks, for external storage, in an energy-efficient manner. The authors refer to this scenario as "monitoring" of sensor networks. PREMON explicitly benefits from existing spatio-temporal characteristics in sensor data. In particular, the authors derive an analogy between snapshots of sensor readings and images based on the fact that spatio-temporal characteristics of data are common among the two. Specifically, they compare pixels in images to individual sensors' readings in snapshots of detailed sensor data and show that sequences of sensor readings are similar to sequential frames of MPEG movies. They then adapt MPEG encoding and algorithms to sensor data. The base-stations in PREMON work as predictors that forecast the set of readings that their sensors are going to sense in the near future. These predictions are represented in concise forms, so called "prediction models", and sent to the sensors. The sensors then transmit their sensed readings only when they are different from the predictions by more than a pre-defined threshold. It is clear that there is a significant overhead at the base-station for computing these prediction models and for sending them to the sensors. The authors argue that this overhead can be reduced by using algorithms that produce high percentage of correct predictions majority of the time, and they show one such algorithm. However, We think that although we assume that base-stations are powerful enough to compute the prediction models, the time overhead cannot be ignored. This clearly restricts the scalability of this approach both in the size of the network (number of sensors) and in the data sampling rate.

3.1.1 Open Problems

Approaches that benefit from spatio-temporal correlation in sensor data face several challenges. First, how to generalize the approach to summary queries, e.g., aggregates? Specifically, are there spetio-temporal correlations among aggregates? Second, how can we learn long and short term patterns in the data online? Nevertheless, can static correlations be learnt "efficiently" online? Can unstable correlations that varies with time be learnt online? We believe that there is a long way to go in order to understand and benefit from correlations in sensor networks.

3.2 Dimensions for Local Storage

The focus of this research is to design Dimension, a system that enable multi-resolution queries on sensor data, while incorporating hierarchical and distributed long term storage. This systems utilizes spatio-temporal correlations between sensor readings by performing *local* compressions on the time dimension then *distributive* compression on the space dimension in order to optimize the overall energy consumption[3]. The Dimension approach can be summarized in the following three steps.

Temporal decomposition In this step, local time-compression processing is performed at each sensor node. This processing is local with no communication overhead.

Spatial Wavelet decomposition The spatial decomposition is performed via a special routing protocol called wavRoute. Data reduction is performed by applying multi-level two-dimensional wavelet transform on the coefficient obtained from the first step, the one-dimensional temporal data, using subband coding. The goals of this step are to minimize communication overhead, balance communication, storage, and computation load among all nodes.

Long term storage Long term storage is performed via aging the wavelet compression progressively over the time. It aims, in general, at enabling spatio-temporal pattern mining.

3.2.1 Open Problems

Dimension is still under development. Once the system is implemented the qualitative benefits of temporal and spatial data compressions can be better studied. Theoretically, better compression does not necessarily translate to better energy savings in sensor networks since both data transmission and passive listening have a cost. The effect of compression on the accuracy of different queries needs extensive investigation. In general, it is not straightforward to quantify compression benefits and to schedule communication in order to obtain energy savings while performing the compression. Also, it is expected that the compression ratio, the total energy savings, and the performance of queries will change based on the application, the network topology, and the nature of the sensed phenomena. However, how can these metrics be measured beforehand? Nevertheless, correlation-learning related problems, similar to PREMON, are still open.

3.3 Geographical Hash Tables for Data Centric Storage

The authors of this approach assume that sensor data can take one of two forms: observations (low-level readings) or events (predefined grouping of data). They also assume that observations are too massive to be directly communicated to outside the network, and therefore, events are defined and queried instead. However, if the users are interested in the low-level observations, they can explicitly extract them from the corresponding node(s). Events are explicitly defined by the users using specific instructions (tasks, e.g., taking readings). The instructions also specify where to store the event upon detection. The key components of the data dissemination approach can be summarized as follows.

GHT: a Geographic Hash Table

Geographic hash tables are used to hash the events (sensed data) to the node where it will be stored. The hashing is computed based on a key associated with each event. The nodes are assumed to know their geographic location, e.g., by using a GPS. The GHT hashes keys into geographic coordinates, and stores a (key, value) pair at the node geographically closest to the hash coordinates of that key, also called the home node. It ensures

[3]Recall that time compression is local, and therefore, is cheaper than space compression that requires communication between sensor nodes.

that the load is distributed evenly throughout the network by using a geographical hierarchy, i.e., when many events map to the same node, *structured replications* are used, in which these events become distributed among multiple mirrors. Two operations are defined to store and retrieve data called Put(k,v) and Get(k), respectively. A suitable routing protocol, called Greedy Perimeter State Routing (GPSR), is used. This protocol has two functionalities: (a) greedy forwarding algorithm that forward packets progressively toward their destination, and (b) perimeter forwarding algorithm, that is based on the right hand rule, for forwarding packets when greedy forwarding is not applicable. Specifically, when there is no node "geographically" closer to the destination than the current node. In this case, the packet traverse the perimeter that encloses the destination, also called the home perimeter, and come back to the home node.

PRP: a Perimeter Refresh Protocol

PRP is a novel protocol that is based on the perimeter forwarding algorithm of GPSR. It provides both persistency and consistency when nodes fail or move. It stores a copy of the (key, value) pair at each node on the home perimeter, i.e., data is replicated locally close to the original home node. It also refresh these copies periodically, which ensures that the copies will be stored at the correct node, i.e., closest to destination, even after node failure or topology changes. PRP generates very low traffic especially in dense networks where perimeters are quite short (3 hops in length). The advantage of this approach lies in its utilization of high local communication which is efficient in dense networks. Nevertheless, scalability in database size and network size are ensured by using a data-centric storage approach.

3.3.1 Open Problems

GHT hashes keys uniformly over the geographic space. When nodes in the network are distributed non-uniformly, the efficiency of the algorithm will definitely decreases due to skewness in storage and forwarding loads. Hence, an open problem is "how can we adapt to such *realistic* situations?". Another aspect is that GHT implicitly assumes foreknowledge of the space boundary and smart nodes that are aware of their geographic locations. It is not clear how to adapt to situations where the boundaries dynamically change or even not known beforehand at all. Also, "What can we do in situations where only approximate node location are provided or when we have no knowledge of locations?". We believe that most of these open problems motivate DCS approaches that are not based on geographical hashing.

3.4 Experimental Evaluations

The major evaluation aspect of any storage approach is the total energy consumption in the network. PRE-MON was implemented and evaluated in real sensor network of very small size (5 sensors). Evaluations showed that the approach can cut down the energy consumption by several orders of magnitude. Dimension, on the other hand, is still under implementation. Simulation results, however, showed that Dimension gives better results in the worst case analysis compared to a fully centralized technique. However, the improvement are not as significant in the average case. The cost metric in this approach so far is either compression-communication tradeoff or compression-error (signal distortion due to lossy compression) tradeoff. Currently, there is no evaluation of compression versus query performance or compression versus computation overhead. The approach still needs extensions to these cases as well as hybrid methods that weights these different parameters (i.e., communication, error, query performance, and computation). For GHT, two metrics were evaluated: (1) the total usage, i.e., the total number of packets sent in the network, and (2) the hotspot usage, i.e., maximum number of packets sent by any particular sensor node. The evaluations showed that GHT is a preferable storage approach in situations where the network size is large (with only one base-station!) and the number of queried events is much less than the total number of stored events. They also showed that if the number of events is large compared to the network size, a local storage approach will be preferable. Finally, the evaluations showed the effectiveness

of the PRP refreshing protocol in offering high data availability, even in node failures and mobility situations. An important critique of Dimension and GHT, however, is that the authors compared the performance of their system against a fully centralized networks with large number of nodes. It looks somewhat clear, and indeed trivial, that the centralized approach will fail in this case. Definitely, in large networks one would divide the network, e.g., geographically, into clusters of nodes of suitable size and use a centralized solution in each sub-network. It will be far more interesting to make a performance study against networks of this structure. For PREMON, a more extensive evaluation of the approach is clearly still needed, using both simulations and real systems, in order to confirm the preliminary results .

4 In-network Aggregation

This section summarizes recent research on online aggregation of sensor streams. Work on this topic is largely due to the USC/ISI and UCLA communities [13, 12, 22, 14, 10]. The research generally focuses on collecting answers to posed aggregate queries by processing the query in the network, in a distributed fashion. This approach differs from centralized processing approaches, i.e., external storage techniques, in that the later collects individual readings at a powerful server and processes any query centrally. Centralized approaches, therefore, are considered costly in some applications where individual raw readings are not important. Furthermore, queries here are assumed to be simple aggregates with a structure similar to aggregates in traditional databases. Hence, they are different from summary queries considered in data-centric techniques. In particular, queries are assumed to be traditional *decomposable* aggregates such as min, max sum, count, average, etc. Decomposable aggregates are those queries that can be evaluated using distributed algorithms. We will discuss this property in more details later in this section. An SQL-like language is also used to define such aggregates queries.

4.1 Two Approaches: An Overview

We consider two approaches to in-network aggregation, TAG [13, 12], a Tiny AGgregation service for ad-hoc sensor networks, and aggregation for monitoring wireless sensor networks [22]. Routing and processing of data cannot be separated in wireless sensor networks. The two approaches, that we will discuss below, share the same technique of in-network processing. They basically differ in how the data is routed in the network and how the answer to the query is collected. Moreover, although the application of each approach seems different, their general objectives are almost identical, i.e., distributed computation of aggregate queries.

Madden *et al.* [12, 13] motivate the need for building systems that provide aggregation in wireless sensor networks as a *core* service. They aim at providing a generic aggregation service in sensor networks in which users express simple aggregation queries from a base-station and the query is then distributed and processed in the network. It uses an SQL-like language with no joins. It also assumes a single append only table called "sensors" with one attribute per sensor input. TAG serve applications that involve remote, difficult to administer, sensors such as monitoring building integrity during earthquakes, habitat monitoring, monitoring temperature and power usage, etc. The authors argue that in such applications, only summaries or aggregates are required rather than the raw sensor data. The objective of Zhao *et al.* [22], on the other hand, is to build a monitoring infrastructure that indicates node failure and other abnormalities of wireless sensor networks, deployed in harsh environments. Their proposed monitoring architecture continuously collects aggregates of different network properties such as number of active nodes, residual energy, loss rate, packet counts, energy levels, etc., in an accurate and efficient way by using decomposable aggregates from the entire network. The architecture detects any sudden change in these properties and, consequently, examines the cause of that change in more details. They provide three levels of monitoring in their approach: Digest, Scans, and Dumps. Each monitoring level consists of a class of tools. In the *Digest* level, the architecture continuously collect aggregates of network properties. In case of a sudden change, the *Scans* tools provide global views of the system state in order to guide system administrator to the location of abnormality. Finally, *Dumps* enable users to collect detailed node state for diagnosis, upon request. Due to similarities between the two approaches, in the rest of this section we

discuss the general technique of in-network aggregation and only highlight the basic differences between the two proposed approaches.

4.2 Query Evaluation

Aggregate queries in in-network aggregation are evaluated in the network using two phases: a distribution phase followed by a collection phase. Only decomposable aggregates such as min/max, sum, average, and count, can be evaluated using such a distributive approach.

Distribution Phase

In this phase the query is distributed to every node in the network. A tree rooted at the base-station is used for data routing, also called the routing tree. Irrelevant data is discarded and only relevant data is combined into a more compact form, i.e., communication cost is reduced in this approach as compared to collecting raw sensor data. The processing continues until the result is finally computed and routed back toward the user. Consider the following illustrative example, shown in Figure 1, where the count of nodes in the network is required. The count query is first flooded to every node in the network starting at the base-station. Each leaf node in the tree reports "1" to its parent. Parents sum counts of their children, add "1", and then reports the result to their parents, and so on. The count, hence, propagates up the routing tree and reaches the root.

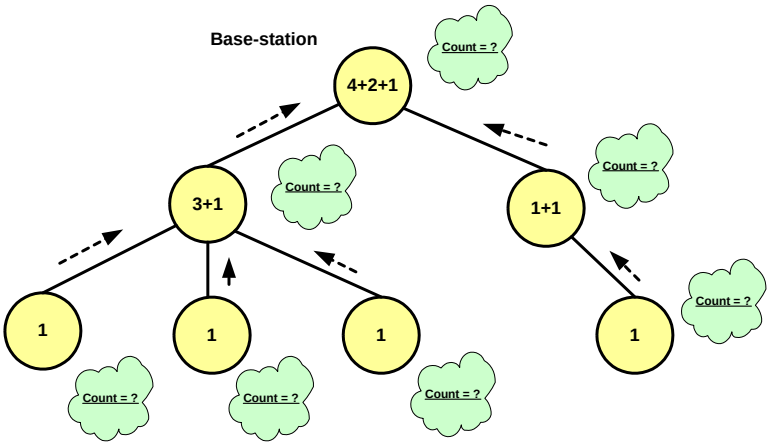

Figure 1. A simple in-network aggregation scenario

Collection Phase

In the collection phase the time to evaluate the query, so called *an epoch*, is subdivided. Parents collect data from children at specific time intervals. These intervals are properly selected to allow collection, combining of partial results, and propagation up the network. Eventually the required aggregate arrives at the root. It is also worth mentioning that similar approach is used for aggregates with grouping. Here, partial aggregates are combined with group id(s) in order to distinguish different groups.

4.3 Routing

Two legitimate questions that arise during query processing are how the routing tree is built and how sensitive this in-network aggregation technique is to node and communication failures. We will defer the answer to the second question till we discuss the performance of the two approaches. The answer to the first question, however, is what distinguishes the two approaches from each other. The TAG approach can use any routing algorithm that provide two functionalities: (1) ability to deliver query requests to every node, and (2) ability to provide one or more routes from every node to the root. In particular, it uses a tree-based routing where one

node is the root. The root periodically broadcasts messages asking sensors to form a routing tree. The message contains the root id and level. When sensors hear the message they assign their own level to the message level + 1, and assign their own parent to be the sender of the message. Sensors then rebroadcast the routing message with their own ids and levels. Children select another parent when parent fails. When specific nodes wishes to send a message to the root it broadcasts the message to its parent, and so on. Zhao *et al.*, on the other hand, propose a routing (propagation) technique, called "Digest Diffusion". This routing technique *implicitly* builds a routing tree and propagates partial results up this tree towards the root to compute the aggregate query. The routing technique does not assume any base-station or a pre-specified hierarchy, rather, it implicitly construct a "digest tree" based on computing either a "min" or a "max" aggregate as follows. Consider the max query, each node i sets its perceived maximum value m_i to its own value, the source of the maximum s_i to i, the hop distance h_i to 0, and periodically sends (m_i, s_i, h_i) to its neighbors. When node i receives a message from its neighbor j with $m_j > m_i$, it sets m_i to m_j, s_i to s_j, h_i to $h_j + 1$, and its parent p_i to j. Node i may switch its parent to k if node k provides the same maximum value but h_k is less than h_j. Gradually all nodes agree on a node s to be the source of the maximum with value equals the reading of s. This technique converges in time proportional to the network diameter. Other aggregates such as average, sum, and count, are computed using this tree, i.e., the tree must exist or be built first using min/max aggregate. The digest tree also adapts to root failure since any node switches to another parent when its parent node fails, also, parents keep response timers for their children, similar to TAG.

4.4 Performance

The authors compare their approaches with a fully centralized approach that has one access point (base-station). Despite the fact that this is neither a fair nor a convincing comparison, we include their performance evaluation for completeness of discussion. As expected, TAG dramatically decreases communication and yields an order of magnitude reduction in communication cost compared to a centralized approach. The same result was reported by Zhao *et al.* for their approach. Another contribution of Zhao's performance evaluation, however, is quantifying the impact of packet loss in sensor networks experimentally. We have already discussed these evaluations in details in Section 2. In general, they showed that heavy packet loss and link asymmetry can be quite common in sensor networks. Also, that this high loss rate and asymmetry can affect the routing tree construction, and in turn, produce significant errors and oscillations in aggregate computation. They showed that different aggregates have different robustness characteristics. For example, min/max queries are the most robust, while count and sum aggregates are sensitive to loss since they rely on partial results from every sensor. The robustness of the average query depends on the distribution of the data, i.e., large uniformly distributed data is more robust compared to skewed distributions. They proposed a low-level networking solution to this loss problem which is based on link quality profiling. In contrast, Madden *et al.* use various techniques, in TAG, in order to improve tolerance to loss and to optimize the communication cost such as: (1) caching the last readings of children, (2) snooping by utilizing the shared radio channels, and (3) guessing, i.e., providing a guess for min/max aggregate so that sensors do not have to send their own values if they do not contribute to the guessed result[4]. They generally benefit from the tight integration of query processing with routing in in-network aggregation. Realistic evaluations of the basic approach of TAG, i.e., in the existence of failures, etc., and without the use of any loss tolerance techniques, showed that TAG is not tolerant to loss. The tolerance improved "slightly" for some of the above mentioned recovery techniques.

4.5 Discussion and Open Problems

In-network aggregation is suitable for specific applications such as monitoring in harsh environments where only summaries or aggregates are required rather than the raw sensor data. However, in-network aggregation cannot easily or efficiently be generalized to other applications with ad-hoc queries or complex queries, or those

[4]The use of guessing bears similarities with PREMON [9].

that require many different aggregate to be computed simultaneously. Nevertheless, history of sensor readings is very useful in many applications where off-line data mining techniques can be applied. Unfortunately, no history of data can be obtained using an in-network processing approach. The overhead of building and maintaining the routing tree should not also be ignored. Moreover, there is a considerable waiting-time overhead, in this in-network technique, till answers become available to the end users. This is due to the hierarchical fashion in which the queries are evaluated. This time overhead scales linearly with the network diameter. All these limitations make in-network aggregation impractical in many applications. The major drawback of the two approaches for in-network aggregation, discussed above, is their attempt to compare their techniques to fully centralized approaches in order to show their superiority. Therefore, more experimental studies need to be performed to fully understand the merits and limitations of in-network processing. It is worth mentioning that the functionality and performance aspects of TAG have been extended recently in [14], where data acquisition issues and their impact on query optimization and execution were discussed. Also, in [10], where sophisticated queries were introduced for the purpose of topographic mapping, wavelet-based compression, and tracking.

Node failures and packet loss are very common in sensor networks. Aggregate computation is, in general, sensitive to loss. This motivates the need for designing general-purpose data cleaning tools for sensor data streams. These tools should not be suitable for data aggregation only, rather, it should be generic and scalable in the number of sensors. This task is very challenging given the need for an online tool and due to the severe limitations of sensor networks.

5 Architectures for Sensor Streams

So far we focused on wireless sensor networks that are basically used for monitoring harsh environments. The posed queries in such deployments were about summaries or detailed sets of sensor data. Furthermore, the major objective was to provide "primitive" mechanisms for gathering sensor data that are energy-efficient. In order to achieve this objective, all approaches, discussed above, were generally application-specific and, clearly, cannot be changed on the fly to suit ad-hoc queries. In this section, on the hand, we discuss research on modeling sensor streams and defining abstractions to represent sensor networks as databases [2, 15, 21]. We also discuss the recent work of Madden *et al.* on designing generic architectures for queries over streaming sensor data [11]. The research discussed in this section aims at constructing sophisticated systems for sensor networks that are not application-specific. These systems enable three types of queries on sensor streams: historical queries, snapshot queries, and long-running queries. Historical queries are defined as aggregates over historical data while snapshot queries specify values of sensor data at a given time instance. Long-running queries are those queries that run continuously over time interval [2].

Traditional database management systems are based on data-pull techniques and an off-line execution of queries. In contrast, sensor data streams consist of massive flow of periodic sensor samples that are "pushed" continuously to the database. Furthermore, queries on sensor data have a real-time nature, i.e., they become useless if their deadlines are missed. Due to this real-time nature and the continuous flow of sensor data, a near real-time processing of such queries is essential. Therefore, traditional database management systems are not suitable for sensor streams and stream management systems are needed instead [11]. However, traditional streaming systems, e.g., Aurora [3], do not take the resource constraints of sensor networks into consideration during query processing. Hence, traditional streaming systems are also inapplicable to sensor streams, and we conclude that handling of sensor streams requires special streaming systems that adapt to the resource limitation of sensor networks. Madden *et al.* argue that existing traditional database management systems handle either streaming data (push-based) or static data (pull-based) but not both. Hence, they propose an architecture for managing multiple queries over streams of sensors and traditional data sources. In other words, a hybrid approach that allows queries that combine streaming push-based sensors with traditional pull-based data sources. Their architecture aims at optimizing the use of sensor resources while maintaining high query throughput. In parallel to this work, the objective of Gerke *et al.*, in Cougar, is to define abstractions that enable representing sensor networks as databases. Specifically, they model each sensor type as an abstract data

type (ADT). They also define semantics for sensor databases where each long-running query is modeled as a persistent view. This view is maintained during the given time interval of that query. In their work, stored data are represented as relations while sensor data are represented as sequences (time series) [2, 15]. In general, the two research directions are considered complementary to each other. In this section, however, we focus on Fjord, the architecture proposed by Madden *et al.* since it is more relevant to our discussion in this survey. We will only highlight some recent relevant contributions of Gehrke *et al.* later in this section.

5.1 Fjording Sensor Streams: An Overview

The architecture consists of two major components: Fjord and sensor proxies. Figure 2 illustrates this basic architecture. Users issue queries using sensor catalogs to the query processor. The query processor processes the query, instantiates operator(s), and looks up corresponding sensor proxy (the specific functionality of the proxy will be discussed shortly). The proxy that controls relevant sensors, instructs them to relay their readings. The sensors relay their readings to the proxy either as raw or processed signals, e.g., aggregate. The proxy packs these samples as tuples and forwards them to the query processor. The processor processes the query using the provided data and, finally, output the answer to the end users.

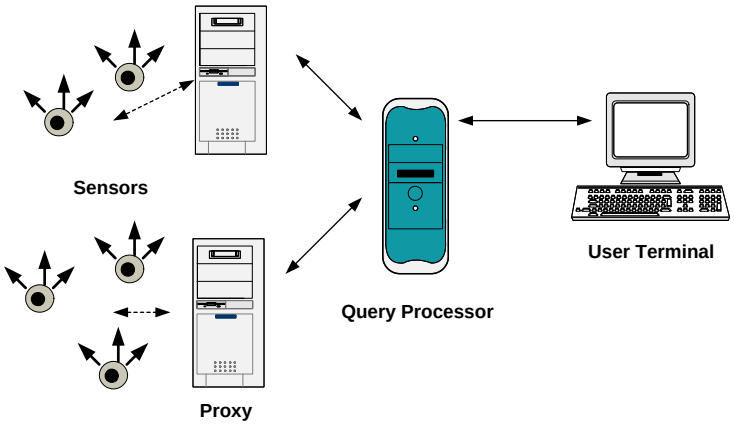

Figure 2. The Overall Architecture of Fjord

Fjord

Fjords, Frameworks in Java for Operators on Remote Data streams, are generalized query plans for sensor streams. They provide non-blocking (passive) and windowed operators that allow queries over both streaming push-based sensors and traditional pull-based data sources. In other words, they integrate streaming sensor data (push-based) with disk-resident data that are pulled by traditional operators (pull-based). Pull-based blocking operators are not suitable for sensor streaming for several reasons. First, sensors cannot keep their receivers on at all times listening to requests for samples due to their resource limitations. Second, sensor networks suffer several problems that make sensor data highly liable to latency and loss. Finally, sensor streams are never ending, and hence, operators on the stream cannot be blocking, e.g., sort and some join algorithms cannot be used. Therefore, operators in Fjord do not pull data from sensors to process, rather, they operate on the sampled pushed data from the sensors.

Fjord provides the functionality and interface necessary to integrate streaming sensor flow into query plans. Specifically, similar to traditional database systems, Fjord consists of operators that export an iterator-like interface and are connected together via pipes or queues (i.e., Fjord = operators + queues). Each operator has a set of input queues and a set of output queues. It reads tuples from the input queues in any order and output tuples to some or all of the output queues. Queues are responsible for routing data from one operator to another. They can either be push or pull queues. Each operator in the query evaluation plan represents a state in a

transition diagram (state machine). Given the current state and some set of inputs, the operator either moves to a new state or remains in the same state and possibly produces a set of tuples as output. This design reduces the total number of threads in the system since all state machine operators can run in a single thread. Unlike other traditional database systems, operators in Fjord do not block if data is not available, rather, they just move back to the same state, without forcing them to output any tuples. Finally, it is worth mentioning that Fjords provide support for combining multiple queries into a single evaluation plan. I.e., they allow processing from multiple queries to share the same data stream and be in a single Fjord. They also support handling operations with multiple inputs and outputs. Experimental evaluations of Fjord showed that this approach is indeed scalable in the number of simultaneous queries. It also revealed that combining related queries into a single Fjord increases the query throughput.

Power sensitive sensor-proxy

The proxy serves as a mediator between the physical sensors and the query processing environment. It uses control messages to switch the sensors on and off and to adjust their sampling rate at any time. This would result in considerable energy savings if users are not interested in querying some specific sensors or even all sensors for long periods of time. Proxies can also control the processing-communication tradeoff by instructing their associated sensors to perform simple aggregation on the samples before transmitting the data. This functionality reduces the load of query processing at the Fjord(s) as well.

5.2 Discussion and Open Problems

The Fjord approach is geared toward sensors with limited resources. Fjord, however, is a centralized architecture and lacks support for advanced query processing techniques. It also lacks optimization tools e.g., that combines common subparts of user queries in order to increase query throughput [11]. In contrast, the early work of Gehrke *et al.* on the Cougar system was geared toward more powerful sensors. Recently, they extended their work and considered the former class of sensors, i.e, sensors with more limited resources. Specifically, they adopted an in-network distributed processing approach into their original Cougar system in order to best utilize sensor resources and introduced novel optimization techniques that suits this class of sensors. Whenever a query is defined, a query optimizer generates an efficient distributed query plan for in-network processing that aims at reducing the resources usage. The catalog information and the query specification are utilized in order to determine the best plan. The generated plan defines the exact steps for executing the defined query such as data flow between sensors, specific computation plan, etc [21]. One important note here is that both Fjord and Cougar have emphasized the importance of proxies as controlling components.

There are numerous open problems and research directions in sensor streaming systems such as the ones discussed in this section. Some of these problems were highlighted by Gehrke *et al.* in [21]. For example, they highlighted the need to overcome limitations of in-network processing, especially synchronization. Such limitations arise from delays and high loss rate in sensor networks as we discussed above in Section 2. Specifically, the design and implementation of adaptive systems, that accommodate sensor delays and/or failure, is an exciting research direction[5]. Moreover, the functionality of the currently used query languages needs to be extended. These languages were not designed for resource constrained data sources such as sensor devices. As we also discussed earlier in this paper, a good query plan on sensor streams is not necessarily the one that only minimizes energy or execution time. In particular, new cost metrics need to be defined that tradeoff various factors. Nevertheless, catalog management related problems need to be solved. Exact and complete metadata about sensors, e.g., their position, workload, etc., is hard to obtain, store centrally, or updated frequently enough. Therefore, new techniques that utilizes only partial metadata are needed.

[5]This functionality is not yet provided neither by Fjord nor Cougar.

6 Erroneous Sensor Data

In this section we give an overview of our ongoing work on handling noisy sensor data [5]. We briefly discussed, in Section 2, that sensor data is expected to be imprecise. The degree of imprecision varies depending on several factors such as sensor cost, environmental effects, etc. Sensor data is basically originating from "actual measurements" of some physical phenomenon and, as such, are subject to several sources of errors. For example, noise from external sources, inaccuracies in the measurement technique, etc. Examples from real-life include, but are not limited to, sensors that record distances to a fixed point by using signal strength. In this case, the estimated ranges can vary widely as signal strength values at the sensor are subject to external conditions. Weights of trucks is another example. They can be measured by means of strain gauges attached to bridges, and therefore, can be affected by other vibrations and calibration drift. Also, the accuracy of temperature sensors depends on their cost, and hence, imprecision in temperature data is highly expected.

The cost of imprecision can be significant if they result in an immediate decision or actuator activation. We believe that sensors will be ubiquitous in the near future, and will become a major source of information for real-time decision making. This motivated our recent work on handling erroneous sensors. This problem does not generally arise in traditional databases where the source of data is either an explicit data entry operation or a transaction activity. The origin of data in this case is typically business, financial or personnel. The database management systems usually assume clean data with no errors, and noisy data, if any, is assumed to be cleaned by an off-line and independent operation prior to storage of data and evaluations of queries. Such off-line operations, however, are usually tedious, lengthy, and require human expertise [16]. Clearly this scenario is not suitable for any data stream. Therefore, in the rest of this section we highlight our recent research on "online" cleaning of noisy sensor streams.

6.1 Online Cleaning: An Overview

The major objective of our current research is to obtain accurate uncertainty models of the true unknown sensor readings in an online fashion. We explicitly incorporate an error model for each of the noisy data sources (sensors). We use the observed noisy readings in conjunction with these error models to provide accurate uncertainty models of the unknown true sensor readings online. The resultant uncertainty models could then be used to estimate real-time queries on sensor data. The uncertainty model can be computed either at the sensor level or at the database server when sensor data arrives. However, models derived at the sensor level may have to simpler than those derived at the powerful database server. This is due to obvious processing limitations of sensors, discussed before. Specifically, we use a Bayesian-based estimation approach to model uncertainty in sensor data due to random errors. We start by obtaining an explicit Gaussian error model for each of the data sources. As sensor data flows, we incorporate the likelihood of obtaining the observed values with prior knowledge of the distribution of the true values and, consequently, we obtain posterior uncertainty models of the unknown true values, i.e., a probability density function of the true unknown value. Different models are obtained at each time instance based on the prior knowledge and the observed values. Our approach is applicable to situations when relatively tight prior knowledge can be obtained. We believe that this is a realistic approach since sensor measurements are usually about a well modelled physical phenomena, and hence, a prior knowledge can easily be obtained.

We argue that models of errors associated with measured sensor data have to be part of any data model in sensor databases. Also, the *estimate operator*, that provides a probability distribution of the measured attribute, should be a fundamental operation in sensor databases. To the best of our knowledge there is no other research that focuses on modeling errors in noisy sensor data and on deriving accurate uncertainty models of such imprecise erroneous sensors. Existing work has assumed the existence of Gaussian uncertainty models of sensor readings (Gaussian pdf). Such models, however, are not accurate since they do not benefit from prior knowledge of the data distribution. The focus of the authors, on the other hand, is to introduce techniques for storing these models as abstract data types (ADTs) and on indexing and retrieving them [6]. Others assumed

uncertainty in sensor databases due to lag of updates, consequently, the data in the database is only an estimate of the actual state. However, this work does not deal with erroneous data and it is not clear how uncertainty due to noise can be incorporated in the current model [4].

In presence of noisy data, queries can only provide an estimate of its true value. We are currently working on several algorithms to evaluate a wide range of queries using our proposed Bayesian uncertainty model. In general, we are investigating algorithms that are not tied to a specific model of uncertainty, and hence, can easily be generalized to other models of uncertainty. We aim at evaluating real-time queries approximately and providing guaranteed error bounds.

6.2 Open Problem

There are several interesting research paths and open problems related to data cleaning in sensor streams. Simply, which data cleaning tasks are necessary/relevant to sensor streams? Irrespective of sensor limitations, can they be performed in one pass? If so, can they be adapted/further approximated to suit limitations of sensor streams? For example, in addition to noise, we believe that missing values are, and will continue to be, a major data cleaning task in sensor streams. Missing values persist even when low-level networking recovery techniques are used [22, 12]. This problem looks even tougher to solve (online) than imprecision due to noise. Missing values may not seem to be an important problem if few of them are expected. However, a more interesting problem is to be able to sample the sensor field for few samples from few sensors and then reconstruct the entire data approximately. This scenario provides much energy/processing savings. Therefore, we are currently extending our research on data cleaning to include this problem. There are also several open problems related to query evaluation over multidimensional imprecise data. Specifically, estimation of multidimensional queries over noisy data. How can we evaluate queries in this case? Which attribute(s) to pick first in order to obtain better confidence? Can we gather additional data in order to improve our confidence? Which data to gather? etc. Finally, another direction is deriving cost metrics for the effect of query estimation on decision making and actuation, e.g., the cost of false positive and/or false negative.

7 Conclusions

We have emphasized several problems and limitations of wireless sensor networks and sensor data streaming. We have also discussed several research directions in sensor streams and highlighted major open problems in these directions. In particular, we surveyed current research in storage and gathering of sensor data, architectures for querying sensor streams, and data cleaning. Sensor networks have certain limitations that make sensor streaming, in general, a challenging task, different from traditional databases and traditional streaming. These limitations should explicitly be addressed during acquisition, storage and querying of sensor data. In general, local processing techniques should be favored whenever possible. Sensor data also have certain characteristics that do not usually exist in traditional data such as its spatio-temporal characteristics. These characteristics can be utilized in order to optimize the usage of sensor resources and overcome some limitations of sensor networks/data. However, experimentations and characterizations is still needed in order to fully understand such characteristics.

We have already listed several challenges in sensor streams throughout this paper. For example, the study of spatio-temporal correlation characteristics, especially dynamic correlation. Also, imprecision due to noise, missing values, massiveness of data, etc., and deriving new optimization metrics in query evaluation. Our ultimate goal is efficient systems that address all these challenges. In the near future, sensors will become very pervasive. They will be embedded in the physical space almost everywhere, collecting overwhelming streams of data. They will behave similar to web-pages or agents posting and/or offering services. A decision making infrastructure will be built on top of these sensors. This infrastructure will enable extraction of any needed information, directly from sensors, on demand, and in real-time. The future of sensor research lies in designing mechanisms that allow this scenario. Specifically, mechanisms for integrating data from huge

number of multiple sources, possibly of different characteristics (sensor type, etc.), cleaning this data, posting it in appropriate forms, and providing answers to millions of simultaneous users' queries in real-time!!

References

[1] AIM Inc., Radio Frequency Identification. http://www.rfid.org/, accessed on April 2003.

[2] P. Bonnet, J. Gehrke, and P. Seshadri. Querying the physical world. *IEEE Personal Communications Magazine, Special issue on Networking the Physical World*, October 2000.

[3] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams - a new class of data management applications. In *Proceedings of 28th VLDB Conference, Hong Kong, China*, September 2002.

[4] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *ACM SIGMOD Conference*, June 2003.

[5] E. Elnahrawy and B. Nath. Cleaning and querying noisy sensors. In *Submitted for review*, 2003.

[6] A. Faradjian, J. E. Gehrke, and P. Bonnet. Gadt: A probability space adt for representing and querying the physical world. In *Proceedings of the 18th International Conference on Data Engineering (ICDE 2002)*, February 2002.

[7] D. Ganesan and D. Estrin. DIMENSIONS: Why do we need a new data handling architecture for sensor networks? In *1st Workshop on Hot Topics in Networks (Hotnets-I)*, October 2002.

[8] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical Report CSD-TR 02-0013, UCLA, February 2002. Submitted for review to INFOCOM 2003.

[9] S. Goel and T. Imielinski. Prediction-based monitoring in sensor networks: Taking lessons from MPEG. *ACM Computer Communication Review*, 31(5), October 2001.

[10] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Towards sophisticated sensing with queries. In *Proceedings of 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, March 2003.

[11] S. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proceedings of ICDE Conference*, 2002.

[12] S. Madden, M. J. Franklin, and J. M. Hellerstein. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *Proceedings of 5th Annual Symposium on operating Systems Design and Implementation (OSDI)*, December 2002.

[13] S. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Workshop on Mobile Computing Systems and Applications*, 2002.

[14] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acqusitional query processor for sensor networks. In *ACM SIGMOD Conference*, June 2003.

[15] P. S. P. Bonnet, J. Gehrke. Towards sensor database systems. In *Proceedings of the Second International Conference on Mobile Data Management*, January 2001.

[16] S. Parsons. Current approaches to handling imperfect information in data and knowledge bases. *Knowledge and Data Engineering*, 8(3):353–372, 1996.

[17] G. Pottie and W. Kaiser. Embedding the internet: Wireless sensor networks. *Communications of the ACM*, 43(5):51–58, May 2000.

[18] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, S. Shenker, L. Yin, and F. Yu. Data-centric storage in sensornets. In *First Workshop on Sensor Networks and Applications (WSNA)*, Atlanta, GA, September 2002.

[19] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A Geographic Hash Table for data-centric storage. In *Proceedings of 1st ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[20] N. Sadagopan, B. Krishnamachari, and A. Helmy. The acquire mechanism for efficient querying in sensor networks. In *The First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA 03)*, May 2003.

[21] Y. Yao and J. E. Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Record*, 31(3), September 2002.

[22] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *1st IEEE International Workshop on Sensor Network Protocols and Applications*, October 2002.