A Study of Approximate Data Management Techniques for Sensor Networks

Adonis Skordylis, Niki Trigoni, Alexandre Guitton

A Study of Approximate Data Management Techniques for Sensor Networks

Adonis Skordylis, Niki Trigoni, and Alexandre Guitton

Birkbeck College, University of London
London, United Kingdom
{adonis,niki,alexandre}@dcs.bbk.ac.uk

Abstract — *Recent developments in sensor network technology have enabled the instrumentation of the physical world with smart devices for monitoring purposes. A large variety of applications could benefit from the pervasive deployment of inexpensive wireless sensor nodes, ranging from environmental monitoring to emergency detection and response. A significant challenge is to prolong the monitoring operation of sensor nodes by efficiently using their limited energy, bandwidth and computation resources. In this paper, we survey approximate data management techniques for sensor networks that exploit the tolerance of most applications to small inaccuracies in the reported data in order to extend the network lifetime.*

1 Introduction

Sensor networks are targeted to a large spectrum of applications, including habitat and environmental monitoring, health monitoring, traffic monitoring, security and surveillance, emergency scenarios and military applications. They consist of a large number of nodes deployed in extended areas over long periods of time. Each node in a sensor network is a small, radio-equipped, battery-powered computer with a number of sensors attached to it, which monitor the ambient environment. A database abstraction of sensor networks has recently gained interest in the sensor network community [1, 2]. Each node can be viewed as a mini-repository of sensor data acquired and stored locally in its memory. The network can therefore be treated as a distributed sensor data management system. Sensor networks differ from traditional distributed databases in that they are limited by stringent processing, bandwidth and, most importantly, energy constraints.

Two ideas that are often used to address the issue of limited resources are aggregation and approximation. In-network aggregation reduces the load of data propagated in the network, by pushing part of the query computation to the sensor nodes. Data streams no longer consist of raw sensor readings, but of partial query results that are progressively merged at intermediate nodes in their way to the base station [3, 1, 2]. Note that in-network aggregation does not compromise the accuracy of query results, but simply distributes query computation to multiple sensor nodes.

In-network lossy compression exploits the fact that a large number of applications can tolerate approximate query results. In large sensor networks, it is not realistic to expect

that query results will accurately reflect the current state of the network. Faults naturally occur in a number of places, for instance during measurement acquisition by sensor devices or in data propagation due to node and link failures. In the light of such uncertainty, users are generally prepared to tolerate various levels of precision for different queries. In this paper, we overview approximate data management techniques for sensor networks that exploit relaxed precision requirements, in order to reduce the communication and sensing load in the network.

1.1 Assumptions and Design Considerations

We start by discussing several assumptions and design aspects that will form the basis for comparing existing techniques:

Data requirements: The techniques developed for approximate data management largely depend on the data requirements presented by the users of the sensor network. We will review techniques that aim at approximating results for different types of approximate queries, including aggregate and non-aggregate queries (e.g. time-series, range queries). We further distinguish approximate queries based on whether they require deterministic or probabilistic error guarantees. In the case of deterministic error guarantees, the estimated query answer must be an ε-approximation of the accurate answer with probability 100%. In the case of probabilistic error guarantees, we introduce a certain degree of confidence in the error of the estimated query answer, i.e. the estimated value of a query answer is within ε of the real value with probability p.

Mode of communication: We distinguish three modes of communication, namely pure *push*, pure *pull* and hybrid *push-pull*. In the *push* mode, nodes are programmed to proactively forward their readings to the base station. In the *pull* mode, users inject short-lived queries at the base station, and these queries are propagated to the sensor nodes in order to pull data on-demand. In the *hybrid push-pull* model, sensors *push* data proactively towards the base station. In case this data is not accurate enough to answer all posed queries, additional information is *pulled* from the network in an on-demand fashion.

Mode of processing: We distinguish between *flat* (*2-tier*) and *hierarchical* (*multi-tier*) network models. In the *2-tier* model, processing can occur either at the source sensor nodes (first tier), or at the base station (second tier). To make this model more general, we also include scenarios where the network is divided into clusters and data is either processed at the cluster heads (first tier) or at the base station (second tier). In the *hierarchical* model, readings from different sensors can be accumulated and partially processed at intermediate nodes, as they are forwarded towards the base station. A typical example is tree-based in-network aggregation of COUNT, SUM, AVG, MIN and MAX queries.

Use of data correlations: Whereas some of the techniques reviewed in this paper make no assumption about the model of the observed physical process, others try to exploit the strong correlations between observed values in order to reduce the sensing and communication overhead. We review techniques that focus on *temporal* and *spatio-temporal* correlations concerning one or more sensor attributes (e.g. correlations between humidity and temperature readings of nearby nodes over a time period).

Adaptive distribution of error bounds: The experts of an application domain define carefully the degree of precision (also referred to as *error bound*) that they are willing to tolerate for their queries. Some of the techniques reviewed in this paper consider the prob-

lem of distributing the input error bound to the individual nodes of the sensor network, whereas others ignore this problem (as irrelevant or out of scope).

Node selection: In many cases, it is possible to answer a query at a required degree of precision by tasking different subsets of sensor nodes. In this paper we compare approximate data management techniques based on whether they are designed to consider alternative query execution plans that activate different sets of sensors. We will review competing approaches to selecting the most efficient plan that minimizes the sensing and communication overhead.

1.2 Paper Organization

We are now in a position to present an overview of recent work on approximate data management for sensor networks, which we organize in three major groups. In the first two groups, we consider related work from a *query processing* viewpoint and distinguish existing approaches based on whether they target aggregate queries or not. In the last group, we study approximate techniques from a *storage management* viewpoint.

1. *Approximate Processing of Aggregate Queries (Section 2):* The popularity of aggregate queries in sensor network applications has motivated a lot of recent work on approximate aggregation techniques. These techniques typically use a hierarchical mode of processing. A few of them try to adaptively distribute the error interval among the source sensor nodes in order to reduce communication. They also vary depending on whether they offer deterministic or probabilistic error guarantees.
2. *Approximate Processing of Non-Aggregate Queries (Section 3):* In many cases, the users are interested in approximations of individual readings of sensor nodes, rather than in aggregated data. For example, habitat monitoring applications are interested in approximate animal tracks, traffic monitoring applications are interested in approximate time-series data measuring the flow of cars in a roundabout etc. Discovering and exploiting the data model behind a physical process has proven extremely beneficial in processing such queries. We discuss compression mechanisms that exploit data correlations and report query answers with deterministic or probabilistic error guarantees.
3. *Approximate Storage Management (Section 4):* Whereas most existing techniques are either pull-based or push-based, we consider separately the interesting features of a handful of hybrid push-pull approaches in Section 3. Most of them are general-purpose, i.e. they are not limited to a specific class of queries. They determine where to store (cache) sensor data in the network and at which level of precision, hence directly influencing the storage management strategy of the sensor network.

2 Approximate Processing of Aggregate Queries

This section focuses on techniques for processing approximate aggregate queries in sensor networks. An aggregate query may use a standard aggregate operator (such as AVG, SUM, MEDIAN, MIN, MAX, COUNT) or a user-defined function. Aggregate operators are characterized according to a set of properties like idempotence, commutativity and distributed computation. An operator allows for distributed computation if, instead of

applying the operator to an entire collection of input values, we can obtain the same result by applying it to separate segments of the input collection and merging the resulting partial results. For instance, MIN, MAX, AVG and COUNT are suitable for distributed computation, whereas MEDIAN is not. In this section we only consider operators that allow distributed computation.

Before considering approximate aggregates, we introduce the main mechanism for processing accurate aggregate queries, referred to as tree-based partial aggregation [3, 1, 2]. In an initial network set-up phase, a tree is constructed using simple flooding algorithms [1], data-centric reinforcement strategies [3] or energy-aware route selection schemes [4, 5]. After a tree is constructed, sensor nodes forward their readings along the paths of the tree, evaluating partial query results at intermediate nodes. Additional communication savings can be achieved by combining the processing of multiple aggregate queries over a fixed communication tree [6].

In certain applications, users may wish to tolerate different degrees of precision for aggregate query results. Instead of letting the network configure the allocation of its limited resources to the current queries in an arbitrary manner, it makes sense to let users control these knobs according to their needs. We present two different approaches to expressing uncertainty in aggregate queries. In the first approach, users define the maximum deviation of the returned aggregate value from the correct answer. If the maximum deviation defined in the query is D and the answer to the query is the scalar value $\bar{v}$, then the real value of the aggregate result must lie in the interval $[\bar{v} - D, \bar{v} + D]$ with probability 1. In the second approach users determine accuracy requirements in probabilistic terms. They enforce a certain degree of accuracy in the estimated result, by defining that the estimated query result $\bar{v}$ should be within D distance from the real value with probability greater than a threshold p.

2.1 Aggregation with Deterministic Error Guaranties

Deligiannakis et al. [7] present an algorithm for processing continuous aggregate queries, which is designed for the aggregation tree model. Each node i in the aggregation tree maintains a *filtering* interval, depicting the range of likely aggregate values at the node. The intervals are set so that the total error at the root is within the desired error budget. If the latest aggregate value at a node lies within its filter, then there is no need to update the parent of the aggregate value, thus reducing communication and conserving energy. Since filters are installed on all network nodes except for the root, the algorithm exploits cases where fluctuations in the values of different sensors cancel each other out at a higher level in the hierarchy, thus avoiding further propagation of updates to the root. Error thresholds are continuously adjusted based on local traffic statistics; the parent evaluates the potential communication savings of increasing the error bounds of its children, and distributes error budgets to maximize the total expected benefit. Additionally, volatile sensor nodes are identified and ignored so that they do not use up most of the available *error budget*.

Yu et al. [8] present a similar hierarchical model for processing approximate aggregate queries with bounded quality constraints. The proposed *Adaptive Approximate Aggregation Protocol* (A^3) is built upon an aggregation tree, whose leaves represent source sensor nodes and intermediate tree nodes represent clusterheads residing on sensor nodes.

Each source node reports its value to its parent and the parent subsequently aggregates all values and reports to the higher level. The parent keeps track of his children's previous values and calculates a *default* value and error bound for each child. As in [7], a child is allowed to communicate only in the case that its value violates the error bound. The algorithms used to adaptively adjust the default values and error bounds of each node are slightly different in the two papers [7, 8].

Cormode et al. [9] use quantile summaries to continuously track the full distribution of values in a highly distributed environment comprised of multiple data streams in remote sites. Their algorithms address large-scale monitoring applications in their generality and are applicable, albeit not specific to, sensor networks. The processing of quantiles differs from previous approaches ([7],[10]) in that it requires monitoring of the entire data distribution. The base station is responsible for answering user queries on the frequency distribution of the union of all streams. The allowed error tolerance is split into two parts ($\varepsilon = \phi + \theta$); each remote site monitors the local stream and if its deviations exceed the threshold defined by θ, the site sends to the base station an updated approximate quantile summary error-bounded by ϕ. In order to minimize the frequency of communication, the proposed scheme uses prediction models piggy-backed to the local quantile summaries. Cormode et al. extended their initial work on flat (2-layer) quantile processing to exploit the merging (cancelling out) of quantile updates in a hierarchical setting. The authors prove that the proposed scheme provides strong $\varepsilon-$approximation guarantees for quantile queries.

2.2 Aggregation with Probabilistic Error Guaranties

Considine et al. [11] and Nath et al. [12] have indepedently developed techniques for approximate aggregation in the presence of packet loss and node failures. When a spanning tree approach is used for aggregation, a single node or link failure results in the loss of an entire subtree of values. Multipath routing has been proposed as an inexpensive solution to address this problem. For duplicate-sensitive aggregates (e.g., SUM and COUNT), however, multipath routing may lead to incorrect calculations. Both [11] and [12] extend the theory of sketches to handle certain duplicate-sensitive aggregates, including SUM and COUNT.

For example, Nath et al. [12] utilize *order and duplicate-insensitive (ODI) synopses*: these are small summaries of partial results that count each sensor's value only once, regardless of how many times or in which order they arrive at a node. The proposed mechanism of *synopsis diffusion* is based on three basic functions which have different implementations for each query type: Synopsis Generation (SG), which outputs a synopsis given a sensor reading, Synopsis Fusion (SF), which combines two synopses into one and Synopsis Evaluation (SE), used on the querying node to translate synopses back to an answer. Continuous queries are assumed and a ring topology is used in order to illustrate the advantages of multipath routing. The proposed techniques in [11] and [12] enable the routing of partial aggregates through multiple paths, and provide fault-tolerant evaluation of approximate aggregates with probabilistic error guarantees.

Hartl et al. [13] further reduce energy consumption in the network by activating a subset of the nodes and use their values to infer the values for the sleeping nodes through Bayesian inference. They specifically focus on the problem of obtaining the average

reading in the monitored field. Bayesian inference uses posterior predictive distributions to arrive to conclusions about data that has not been observed, given the existing observations. The algorithm developed is called *infer* and consists of two phases: during the first phase, a distributed algorithm decides which nodes are going to be active or asleep for the next epoch, which contains several sensing rounds. Both node energy level and the values of sensed data are considered while making this decision, in order to achieve load balancing while making sure that important data will always be available to the sink. The second phase of *infer* concerns only the sink: it utilizes Bayesian inference in order to predict the average sensor reading of the network, based on the available incomplete dataset. The Bayesian inference approach has the potential of being applied to process a large spectrum of queries beyond aggregates.

3 Approximate Processing of Non-Aggregate Queries

Sensor networks offer new opportunities to monitor the real world at an unprecedented scale, since they have the potential of covering extended areas for large periods of time. From a research viewpoint, this becomes a real challenge when the application requires fine data granularity instead of data summaries. In this section, we study techniques that aim at processing non-aggregate queries in an energy-efficient manner. They rely on temporal and spatio-temporal correlations in the observed data to significantly reduce communication and increase network lifetime. In Section 3.1, we discuss techniques aimed at minimizing communication from a single sensor, and in Section 3.2, we extend our discussion to minimizing communication from multiple sensors.

3.1 Approximate Processing of One Sensor Stream

In [14], Lazaridis et al. present techniques for compressing time series values produced by sensor nodes. Their approach focuses on capturing the complete history of a time series, arguing that knowledge of the way the quantity being monitored evolves is essential for a variety of scientific applications. They assume a network of multiple sensors, the *producers* and a central database, the *archiver*. The goal is to capture an approximate version of the time series on the archiver, with quality guarantees for each individual value, and to predict future values of the series based on this approximation. The authors utilizes a piecewise constant approximation (*PCA*) algorithm. PCA is a lossy compression scheme that represents a time series as a sequence of value-interval pairs called segments (c_i, e_i), i.e. a constant value c_i during e_i epochs. They use Poor Man's Compression (*PMC*), an online version of PCA that computes PCA segments as each value arrives. For example, PCA-Mean samples values and computes their mean, until the mean differs from the minimum and maximum by more than a given threshold. The time required to compute a segment at the producer and send it to the archiver may be long if the time series consists of similar values. To address this problem, the authors consider embedding a prediction model to answer queries concerning current and future values that have not yet been covered by a segment. The model's precision interval can be adjusted using techniques similar to the ones described in [10].

Earlier work on processing distributed data streams proposed the caching of data intervals at the source and the base station, and suggested that the source should refrain from

propagating its values as long as they fall within the cached interval [10]. Jain et al. [15] identified shortcomings in the approach of caching static data to reduce communication between the source and the base station. They propose the use of Kalman filters, and instead of caching data intervals they cache filter parameters that help predict the data. A Kalman filter is a recursive algorithm that predicts the future internal state of a system based on current state information and observations of its external behaviour. The recursive procedure is based on two mechanisms: *Prediction*, which estimates the system state at time $k+1$ and *Correction*, which adapts the filter according to the actual values observed at time $k+1$, making the filter a truly *online* adaptive model. The authors propose the *Dual Kalman Filter Model*, an architecture similar to the one described in [9]. Continuous queries are directed to a central server, accompanied by a precision constraint. For a query concerning node i, a Kalman filter KFi is installed on the server and a mirror of this filter is initiated on source i, simulating KFi's operation. The server uses the predictions of the Kalman filter to provide answers about i in the future. Updated values from sensor i are propagated to the server only when the mirror filter (and therefore KFi as well) is found to be incapable of predicting i's values within the specified constraint. The Kalman filter is able to model various processes, including non-linear and noisy ones, making it feasible to apply the same framework to a variety of scenarios.

In recent work, Deligiannakis et al. [16] present a new technique designed for historical data compression in sensor networks. They assume that each source node produces multiple streams of data, one for each type of sensor, and pushes them to a base station. They first construct a *base signal* from the node's time series and subsequently use it to approximate time series using regression. The authors' intention is to exploit correlations between different time series on a node. The algorithm presented is called *Self-Based Regression (SBR)*: it is given a bandwidth constraint (maximum packet size) and the size of the base signal as inputs and tries to provide an optimal approximation to a node's time series with respect to these constraints. The base signal is initially constructed; SBR splits the locally-stored part of the time series into intervals of various sizes and for each one of them computes the benefit of approximating the rest. The interval that incurs the highest benefit is chosen as a part of the base signal and the process is repeated until the desired base signal size has been reached. The next step is to approximate the series data using the base signal; the algorithm concatenates data from all time series in a node, breaks the resulting stream into intervals and tries to approximate them using equal-sized base signal intervals. The error metric used is the sum squared error and it can be modified to any other appropriate metric. The algorithm was shown to outperform several approximation techniques, including Wavelets, Histograms and Discrete Cosine Transform (DCT), using various real-world datasets.

3.2 Approximate Processing of Multiple Spatially-Distributed Sensor Streams

In this section we present recent techniques for processing multiple sensor streams in an energy efficient manner. These techniques make use of both spatio-temporal correlations to perform data reduction.

Guestrin et al. [17] propose an algorithm that manages spatio-temporal data in a distributed fashion. The network is assumed to contain multiple overlapping regions and in each region significant spatial correlations are expected to be observed. The authors

introduce the usage of *kernel functions*; a region's kernel function maps a point x to a number, depending on the position of x in the region. Basis functions are chosen for each region and the kernel functions are used to weigh these basis functions accordingly during the regression process. Kernel regression is based upon a distributed version of Gaussian elimination which computes the weights for each basis function resulting in the kernel regression estimate that minimizes the mean square error for all the data in a region. A query specifies a set of kernels (regions) and a set of basis functions (e.g. polynomial with maximum degree 3) for each kernel, the parameters of which are computed and returned by the system. Neighbouring nodes need to exchange messages containing basis function coefficients in order to optimize the regression estimate. Regression was observed to perform better as the number of basis functions is increased, especially at times when values change dramatically.

Deshpande et al. [18] propose the use of statistical models of real world processes to reduce the cost of sensing and communication in sensor networks. The prototype built, *BBQ*, consists of a declarative query processor and an underlying probabilistic model and planner, based on time-varying multivariate Gaussians. A multivariate Gaussian is the n-dimensional extension of the simple probability density function (*pdf*), represented by means and covariances. The initial representation of the pdf is constructed during a training phase, with the use of historical data. The model runs at the base station, where users submit SQL-like queries that include error intervals and confidence bounds. Multiple one-shot queries are considered rather that continuous ones. The pdf is used to determine whether the model is able to answer a query within the target error and confidence bounds without requiring updated values from the network. If not, the query processor has to pick the best set of attributes to observe (called an *observation plan*) in order to bring the confidence up to the required threshold, while minimizing the sensing and communication cost. BBQ uses a greedy incremental heuristic to identify good observation plans in order to answer range, value and aggregate queries.

Chu et al. [19] tackle the `SELECT *` problem for sensor networks. They propose a framework for selecting approximate data with bounded-loss guarantees, while minimizing communication costs throughout the network. The prototype built, *Ken*, utilizes a pair of dynamic probabilistic models, one running at the base station (*sink*) and the other one at each node in the network. Ken bears several similarities to BBQ proposed by Deshpande et al. [18] as it evaluates a probability density function and exploits both spatial and temporal correlations between sensors. Their key difference lies in the fact that BBQ is pull-based, i.e. the base station acquires data from the network only when necessary in order to satisfy queries, while Ken is *push-based*: source nodes monitor their data continuously and only route values towards the sink upon detecting that the sink's model needs to be synchronized with the local ones. In effect, Ken addresses outlier detection, the basic weakness of BBQ. In order to utilize spatial correlations, Ken partitions the sources into clusters, called *cliques*, and inference is done on one node per clique, the *clique root*. Finding an optimal partitioning scheme for the network is an NP-hard problem so the authors provide a greedy heuristic that runs locally at the sensor nodes.

4 Approximate Storage Management

The techniques discussed in Sections 2 and 3 are either push-based or pull-based. In the push mode, data is proactively forwarded from the sensor nodes to the base station, whereas in the pull mode, data is acquired on-demand only when users inject queries into the network. The push mode is suitable for classes of applications that require continuous monitoring at a constant level of precision. Results are forwarded to the base station periodically or when an interesting event occurs. The pull mode is suitable for classes of applications where monitoring is not needed continuously and throughtout the network area; users require data infrequently each time with a potentially different precision constraint. For a variety of applications that lie between the two extremes, neither the push nor the pull mode allow for energy-efficient data dissemination.

In this section, we discuss hybrid push-pull data management techniques that rely on the storage capabilities of sensor nodes. They allow the distributed caching of sensor data within the network at different levels of precision, and raise a number of interesting research problems: i) adaptive precision setting of sensor data, and ii) efficient aging of sensor data and iii) energy-efficient search and access of stored data.

In [10], Olston et al. focus on the first problem of adaptively adjusting the precision of cached approximations to achieve optimal performance in a dynamic environment. They assume that an accurate data representation (e.g. an integer value) is stored locally at the source sensor node, and an approximate data representation (e.g. an interval) is forwarded and stored at one or more cache nodes where queries are injected. If the source detects that its value has drifted outside the interval, it initiates a transmission to update the cache, as in [7] and [8]. However, unlike [7] and [8], the approximate data stored in the cache is not always accurate enough to answer all of the injected queries. If a query's precision is higher that that offered by the cache, the query pulls the exact value from the source node. To avoid source-initiated refreshes, interval widths have to be increased so that the values remain within them. On the other hand, to avoid query-initiated refreshes interval widths have to be decreased, so that adequate precision is offered for most queries. Olston et al. propose an algorithm to search for the interval width that minimizes the probability of a refresh happening, and compare it with competing approaches like Divergence Caching [20].

In [21], Ganesan et al. focus on the problems of graceful aging and multi-resolution storage and search in sensor networks. They use *wavelet* compression to summarize the data in various resolutions and store the summaries in a hierarchical manner along a quad tree. At higher levels (near the root) summaries are highly compressed but correspond to a larger set of the network data. Querying is performed using a technique that is common in data-mining, called drill-down querying; each query is injected at the highest level, where only coarse summaries exist. Queries that ask for low-precision data are processed locally and the remaining ones are forwarded further to nodes that carry the relevant data at higher resolutions. In order to cope with limited storage, a parametric *progressive data aging strategy* is proposed, which determines how long each node must maintain summaries at various resolutions before discarding them. If training data is available, the aging function is tuned to maximize query quality, otherwise the authors propose the use of a greedy aging strategy.

In their recent work, Wu et al. [22] consider a push-pull approach to executing various types of one-shot approximate queries with bounded quality. They propose a *two-tier* data storage scheme, in which the base station stores an imprecise version of sensor data, and the sensor nodes retain their high-quality readings in local memory. Similarly to [7] and [8], sensor nodes initiate communication with their parents in the routing tree only when their values drift outside an approximation interval, defined by a specific error margin. The values are propagated all the way to the base node, which stores them together with their approximation intervals. When a query is administered at the base station, it is answered at no cost if its error tolerance exceeds that of the centrally stored data. Otherwise, the base station carefully selects which nodes to query in order to answer the query with minimum effort. Wu et al. present node selection (*refresh*) strategies for different types of queries, including top-k, range queries, id queries and aggregates. For example, for aggregate queries, the proposed algorithm aims at reducing the load of hot-spot nodes near the base station by favouring nodes in the same subtree. In addition, the algorithm prefers nodes with high energy levels and small hop distances from the base station.

5 Conclusions

Research in approximate data management for sensor networks is motivated by the fact that most applications do not require accurate reports of sensor data - as long as the data comes with deterministic or probabilistic error guarantees. The flurry of research in this area shows that there exists a great potential in trading the accuracy of query answers for energy savings and increased network lifetime. The techniques discussed in this paper make different assumptions about the mode of communication (flat vs. hierarchical) and computation (pure push, pure pull, or hybrid push-pull); they target a variety of query types, and focus on different research challenges (like adaptive adjustment of error bounds and node selection). They typically exploit correlations between the sensor values, in order to reduce the sensing and communication cost in the network. The different aspects of recent research efforts are summarized in Table 1.

In the near future, sensor networks are expected to be incorporated in a variety of domains, which will raise new application-specific challenges for approximate data management. As the number of competing techniques increases, an issue that we will need to address is how to identify the candidate that best fits the requirements and constraints of a specific sensor deployment.

Acknowledgments

Our work was supported by the Engineering and Physical Sciences Research Council (EPSRC) under the TIME-EACM award EP/C547640/1.

References

[1] Y. Yao and J. Gehrke. The Cougar approach to in-network query processing in sensor networks. *SIGMOD record*, 31(3):9–18, September 2002.

[2] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

Paper	Data requirements (aggr. vs non-aggr. / determ. vs probab.)	Communication (pure vs. hybrid)	Processing (flat vs hierarch.)	Correlations (temporal vs spatio-temp.)
Deligiannakis [7]	aggr. determ.	pure	hierarch.	temporal
Yu [8]	aggr. determ.	pure	hierarch.	temporal
Cormode [9]	aggr. determ.	pure	both	temporal
Nath [12]	aggr. probab.	pure	hierarch.	
Hartl [13]	aggr. probab.	pure	hierarch.	spatio-temp.
Lazaridis [14]	non-aggr. determ.	pure	flat	temporal
Jain [15]	non-aggr. determ.	pure	flat	temporal
Deligiannakis [16]	non-aggr. determ.	pure	flat	temporal
Guestrin [17]	general	pure	hierarch.	spatio-temp.
Deshpande [18]	general probab.	pure		general
Chu [19]	general determ.	pure		spatio-temp.
Wu [22]	general determ.	hybrid	flat	temporal
Olston [10]	non-aggr. determ.	hybrid		temporal
Ganesan [21]	non-aggr. determ.	hybrid	hierarch.	spatio-temp.

Table 1: Design aspects of recent approximate data management techniques.

[3] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MOBICOM*, pages 56–67. ACM, 2000.

[4] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *MOBICOM*, pages 181–190. ACM, 1998.

[5] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report UCLA/CSD-TR-01-0023, University of Southern California, 2001.

[6] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman. Multi-query optimization for sensor networks. In *DCOSS*, 2005.

[7] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Hierarchical in-network data aggregation with quality guarantees. In *EDBT*, volume 2992 of *LNCS*, pages 658–675. Springer, 2004.

[8] X. Yu, S. Mehrotra, N. Venkatasubramanian, and W. Yang. Approximate monitoring in wireless sensor networks. Technical Report 382, UCI ICS, 2004.

[9] G. Cormode, M. N. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: distributed tracking of approximate quantiles. In *SIGMOD Int. Conf. on Management of Data*, pages 25–36. ACM, 2005.

[10] C. Olston, B. Thau Loo, and J. Widom. Adaptive precision setting for cached approximate values. In Timos Sellis and Sharad Mehrotra, editors, *SIGMOD Int. Conf. on Management of Data*, pages 355–366. ACM, 2001.

[11] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *ICDE*, 2004.

[12] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *SenSys*, pages 250–262. ACM, 2004.

[13] G. Hartl and B. Li. Infer: A bayesian inference approach towards energy efficient data collection in dense sensor networks. In *ICDCS*, pages 371–380. IEEE Computer Society, 2005.

[14] I. Lazaridis and S. Mehrotra. Capturing sensor-generated time series with quality guarantees. In *ICDE*, pages 429–442. IEEE, 2003.

[15] A. Jain, E. Y. Chang, and Y.-F. Wang. Adaptive stream resource management using Kalman filters. In *SIGMOD Int. Conf. on Management of Data*, pages 11–22. ACM, 2004.

[16] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing historical information in sensor networks. In *SIGMOD Int. Conf. on Management of Data*, pages 527–538. ACM, 2004.

[17] C. Guestrin, P. Bodi, R. Thibau, M. Paski, and S. Madden. Distributed regression: an efficient framework for modeling sensor network data. In *IPSN*, pages 1–10. ACM, April 26–27 2004.

[18] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, pages 588–599. Morgan Kaufmann, 2004.

[19] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *ICDE*. IEEE, 2006.

[20] Y. Huang, R. H. Sloan, and O. Wolfson. Divergence caching in client-server architectures. In *PDIS*, pages 131–139. IEEE, 1994.

[21] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. In *SenSys*, pages 89–102. ACM, 2003.

[22] M. Wu, J. Xu, and X. Tang. Processing precision-constrained approximate queries in wireless sensor networks. In *MDM*, May 2006.