MASTAQ: A Middleware Architecture for Sensor Applications with Statistical Quality Constraints

Inseok Hwang
EECS Dept, KAIST
inseok@nclab.kaist.ac.kr

Qi Han
CS Dept., UC Irvine
qhan@ics.uci.edu

Archan Misra
IBM Research
archan@us.ibm.com

Abstract

We present the design goals and functional components of MASTAQ, a data management middleware for pervasive applications that utilize sensor data. MASTAQ allows applications to specify their Quality-of Information (QoI) preferences (in terms of statistical metrics over the data) independent of the underlying network topology. It then achieves energy efficiency by adaptively activating and querying only the subset of sensor nodes needed to meet the target QoI bounds. We also present a closed-loop feedback mechanism based on broadcasting of activation probabilities, which allows MASTAQ to activate the appropriate number of sensors without requiring any inter-sensor coordination or knowledge of the actual deployment.

1. Introduction

Sensor networks are currently *data-rich,* but *method-poor,* with relatively few programming APIs that provide higher-level and topology-independent data-management abstractions for pervasive computing applications. In this paper, we introduce MASTAQ, a data management middleware that we are building to provide an easy-to-use and energy-efficient data query API over sensor networks. MASTAQ is based on two fundamental observations on a class of sensor network applications:

- Many sensor applications aim to detect environmental "events" or "states", where the states are computed over *readings from a set of sensor nodes*, rather than individual values. Accordingly, the query objectives of sensor applications can often be expressed in terms of *statistical Quality-of Information* (QoI) parameters, such as the tolerable thresholds on "false-alarm" or "missed-event" probabilities, or bounds on statistical metrics (such as standard deviation) of the reported data set.
- We expect many sensor network deployments to have *significant redundancy in ambient conditions*, with a small fraction of nodes often providing enough data to satisfy the QoI bounds. By activating additional nodes only when necessary, the operational lifetime of the deployed sensor nodes can be significantly extended.

Sensor environments present a tradeoff between the (statistical) accuracy of the state estimated from the sensor readings, and the energy consumption of the sensor network. In many pervasive applications, the accuracy or resolution desired from the sensing substrate changes, depending on the application's context or event history. Moreover, we shall see that expressing accuracy objectives via statistical metrics naturally translates into spatial sampling resolutions that vary with changes in the physical state. MASTAQ's aim is thus to provide a "tuning knob" by which applications can operate at various points on a hypothetical "accuracy vs. energy consumption" curve shown in Figure 1. In many instances, the number (or density) of sensor nodes needed to meet a target QoI objective is *not explicit, but depends implicitly on the values reported by the sensors.* In other words, the behavior of the underlying physical environment (such as the temporal or spatial "frequency" components) dynamically dictates the amount of sensor resources activated. Accordingly, one of MASTAQ's major innovations is the use of a closed-loop controller within the middleware, which adaptively adjusts the level of activated sensor resources to match the target QoI bounds. This paper primarily presents the design goals and architectural components of MASTAQ, and subsequently details a specific closed-loop control mechanism. This notion of using statistical QoI values to retrieve or manipulate varying numbers of data sources is quite distinct from related recent work in the area of sensor databases and statistical queries. For example, while TRAPP [8] supports energy efficiency by retrieving data values from different sources intelligently to appropriately bound the error in an answer (such as the mean value), it does not adjust the set of active data sources itself based on some statistical QoI measure. Similarly, the BBQ approach [2] promotes energy efficiency by using a training sequence of data from all sensor nodes to first compute the parameters of a Gaussian correlation model across *all* sensor nodes, and then uses the model to optimize the number of sensors that are explicitly polled to satisfy the statistical bounds of a query. The intent here is to learn and predict the long-term correlation among sensor values under normal operating conditions, rather than adjust the sampling resolution in the face of unexpected events. In contrast, we explicitly aim to save the network communication cost by activating and retrieving data from additional nodes only when the observed QoI from the sampled data set is not good enough.

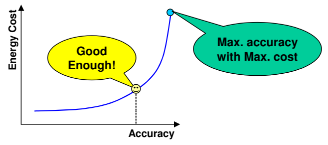


Figure 1. The Accuracy-Energy Tradeoff Model in Environmental Monitoring

To further intuitively understand the design goals of MASTAQ, consider a hypothetical "environmental monitoring" application that uses "smoke" and "temperature" sensors deployed over a large geographical area (e.g., the state of California) to detect the occurrence of "forest fires". Also, imagine a dense deployment, with 100 sensors of each kind deployed uniformly over every 1 square-mile grid, monitoring a total "field" of 500X500 mile, that aims to provide an "early warning" of potential fires in each individual 1-square-mile grid. Clearly, in normal conditions, activating say, only 10, temperature sensors in each grid should prove sufficient–if 10 "randomly" selected sensors in a grid all indicate normal temperatures, a forest fire is "unlikely" to be raging! Of course, if the data from these sensors indicate some anomaly in a particular grid, then the application can activate more sensors (say 50) in the grid to get a more fine-grained estimate of the grid's state. The main insight is that any set of sensor readings imply a certain statistical accuracy over the event-detection process. In this particular example, using a measure such as the "variance" or the "median" reading from 10 sensor nodes should intuitively provide a fairly good *likelihood* of fire detection. While using a similar measure over all 100 nodes would increase the confidence of the answer, it would sharply increase the sensing and communication energy overheads.

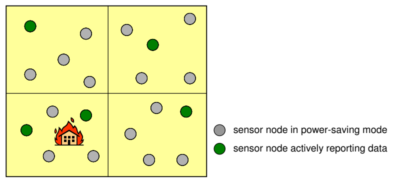
The rest of the paper is as follows. Section 2 explains the general types of statistical queries and presents our initial set of supported queries. Section 3 presents the principal functional components of MASTAQ that we have developed so far. Section 4 then presents MASTAQ's innovative "closed loop control" technique for activating the appropriate number of sensors over any physical grid. Finally, Section 5 concludes the paper.

2. Utility of Queries supporting Statistical QoI

MASTAQ exposes a QoI-centric query model to sensor applications. Broadly speaking, an application defines its QoI requirements either in terms of a) bounds on the estimation/detection accuracy, or b) directly in terms of statistical bounds on the observed data. There is a very active body of state detection research (e.g. [7]) investigating bounds on "false alarms" or "missed detection" probabilities as a function of the sensor node density and layout characteristics. In general, these models are "parametric", deriving these estimates based on a-priori models of the field's behavior or the error model (e.g., Gaussian) on each data sample. Clearly, such models can also be used to answer the dual question: given a-priori statistics and a target detection probability (e.g., $\leq 5\%$ false alarms for a fire that is ≥ 0.2 square-miles), what is the number (or density) of sensor nodes needed? Alternatively, in the absence of explicit models on either the physical phenomena or individual node behavior, applications can specify "non-parametric" queries as predicates over certain statistical metrics over the raw, observed data. For example, a monitoring application can indicate that it wishes to receive the "average temperature reading over region X, with the average computed over a set of distinct sensors that's large enough to assure that $\sigma \leq 5$", where σ indicates the standard deviation.

MASTAQ shall eventually support both these forms of QoI specification. However, the middleware is not directly concerned with the building of statistical models for parametric QoI estimation (work done for example in [2])–rather, as we shall see in the next section, MASTAQ utilizes a library of externally-developed algorithms to convert appropriate application "estimation/detection" preferences into lower-level requirements on node density. Since useful models of parametric "collaborative estimation" are still being researched, our first prototype of MASTAQ seeks to support only the non-parametric "data-oriented" QoI model. Thus, applications specify their requirements over metrics such as the mean, median, standard deviation, percentiles and confidence intervals. An application query does not directly specify the number of distinct sensors; rather, MASTAQ adaptively computes the sample size (number of sources) needed to satisfy the QoI bounds.

It is natural to ask what sort of applications can benefit from such "statistically significant" answers. To answer this, note first that summarization operators (such as average or median) compress the state of particular regions into a single value, something that is often useful when performing coarse monitoring over large spatio-temporal regions. For example, an application monitoring the entire state of California would ideally like to display an "average" temperature number for each zip-code, abstracted from the 1000s of sensors in each zip-code. Of course, if a zip-code (say "90200") displays an abnormally high value, the application would reissue a query for "90200" with higher QoI, viewing averages computed at, say, street-level granularity.



sensor node in power-saving mode
sensor node actively reporting data

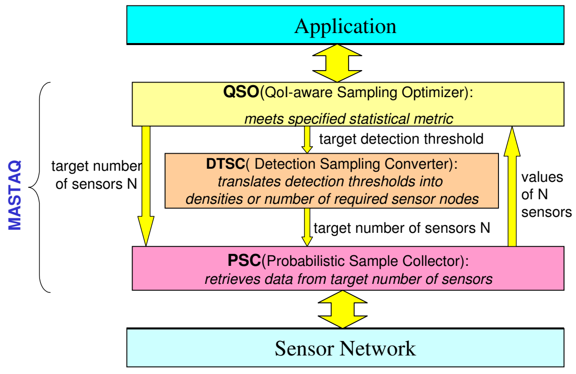
Figure 2. An Example of Statistical Monitoring of a Forest Fire

Second, bounds on statistical measures such as variance naturally translate into spatial sampling resolutions that change as the underlying physical state changes. For example, if 10 sensors, sampled at random in a particular zip-code, all indicate temperatures in the range (73,75), intuitively, the entire zip-code is likely to have a relatively small temperature variation. Conversely, a wide temperature variation among the 10 sensors implies either a discontinuous physical phenomena (one sub-region has a "small fire", while other portions are normal) or the selection of error-prone sensors (in which case, additional sensors should provide better confidence).

Of course, using a set of active sensors that are relatively spatially sparse runs the risk of missing events that might occur "locally". For example, in Figure 2, even though the 2 selected sensors indicate "normal" temperature, there is a "fire" event that has been missed even though one of the 3 inactive sensors already has a "hot" reading. Fundamentally, we observe that a lower QoI (reduced spatial resolution) delays the detection of an event (the fire), until it becomes large enough. In many instances, this delayed detection is acceptable, and justified by the significantly lower energy cost of operating the sensor network. Indeed, many practical applications are only interested in persistent events that eventually grow to be "large enough". (Thus, subsampling misses the small brush fires that burn themselves out; however, these are of lesser concern. Once the fire is detected, however, using the entire set of available sensors to obtain a very-detailed thermal map is very useful in modeling the subsequent spread of the fire.) MASTAQ may thus not be useful for applications such as intrusion detection which need to track transient events. However, it is particularly appropriate for environmental monitoring applications (e.g., oil-spill detection, hurricane detection, etc.), which do not have hard real-time constraints on detection latency and can thus benefit from the "accuracy-energy" tradeoff of Figure 1.

3. Principal MASTAQ Components

To support statistical queries from applications, MASTAQ consists of three components (Figure 3): *QSO* (QoI-aware Sampling Optimizer), *DTSC* (Detection-to Sampling Convertor) and *PSC* (Probabilistic Sample Collector). The DTSC is a library of externally-developed algorithms that translate estimation/detection thresholds into densities or number of required sensor nodes. This translation may either be based on a-priori models, or employ learning algorithms on "training data" to compute the parameters of a specific model (as in [2]). Similarly, the PSC may be a library of techniques by which MASTAQ activates a specified set of resources in the sensor network. The QSO interfaces with applications, calling the DTSC if the application query is specified in terms of detection thresholds. If the application QoI preferences are "non-parametric" (directly expressed over the set of sensor data), the QSO directs the PSC to obtain the data samples from the desired number of sources. The QSO then dynamically monitors the statistical metric to ensure that the set of data samples (each sample from a distinct source) meets the QoI bound. Accordingly, the chosen PSC component interacts directly with sensors and obtains readings from the target number of sensors, as requested by the QSO.



Application
QSO(QoI-aware Sampling Optimizer):
meets specified statistical metric
target detection threshold
MASTAQ
target number of sensors N
DTSC(Detection Sampling Converter):
translates detection thresholds into densities or number of required sensor nodes
values of N sensors
target number of sensors N
PSC(Probabilistic Sample Collector):
retrieves data from target number of sensors
Sensor Network

Figure 3. The Architecture of MASTAQ

The application query to MASTAQ specifies the "type" of sensor data desired, the spatial region and time duration over which the data is to be collected, the "function" on the underlying data set that is to be returned (such as the sum, median or average), and predicates on the QoI of the data set. The QoI predicates can refer to both statistical (e.g., the standard deviation) and deterministic (e.g., the sampling period) metrics. While we do not discuss the entire query structure (which is really a long-lived subscription) due to space limitations, Figure 4 provides a sample XML representation of a sample query: compute the average temperature reading over the rectangular grid (100,200,500,500), once every 2 minutes, with at least 5 distinct sensors, such that $\sigma \leq 5$. In our initial implementation, we will support

```
<statistical-query>
    <type>temperature</type>
    <region>(100,200,500,500)</region>
    <duration>600</duration>
    <function> average(temp value)</function>
    <predicate>σ < 5 & period=2 & minsamp=5</predicate>
</statistical-query>
```

Figure 4. A sample application query expressed in XML

predicates over two different statistical metrics: the "standard deviation" of all the samples and the "confidence level" of the returned value (as measured by the batch means analysis technique).

QSO adaptively translates each statistical query into the number of samples needed N. Assume that a query requires the standard deviation of all the collected samples σ to be lower than a threshold θ: $\sigma \leq \theta$. To meet this requirement, simply increasing N does not necessarily decrease σ. A straightforward strategy of adjusting N makes its decisions based on (a) whether $\sigma \leq \theta$ or $\sigma > \theta$ currently; (b) whether N is increased or decreased in the previous period; and (c) whether σ is increased or decreased due to the change of N in the previous period. In addition, since getting more samples always means consuming more energy, N is increased more conservatively comparing to decreasing N. For example, assuming that $N = 100$ at period i with $\sigma \leq \theta$, we decrease N by half to 50 at period $i + 1$, which leads to $\sigma > \theta$, so at period $i + 2$, we increase N to $50 + \frac{50}{4} = 62$. Alternatively, information theory may be used [6] to decide a sampling frequency such that the region of interest can be reconstructed. We defer the discussion of ongoing research in QSO adaptation techniques to a future paper.

4. Design and Evaluation of A Probabilistic Sample Collector (PSC) Technique

While the QSO adaptively estimates N, the number of distinct samples (distinct activated sensors) needed in a particular region to achieve the target QoI bound, the PSC is responsible for actually obtaining the N readings. In this section, we describe a controller-based implementation of the PSC that possesses two important properties:

- It does not require MASTAQ to be aware of the identity of individual sensor nodes, or even the total number of available nodes. As a consequence, nodes may be added or removed (or fail) from the underlying infrastructure, without any coordination with the PSC.
- It does not require an individual sensor node to be aware of, or interact with, any other sensor node. It thus avoids the complications and overhead of inter-sensor coordination techniques.

Of course, MASTAQ may include other alternative externally-developed implementations of the PSC.

To avoid these two requirements, our approach employs a broadcast-based feedback control mechanism. In the first step, the PSC broadcasts an "activation request" (or request for data) to all nodes in a designated geographic region. (Given the advances in localization algorithms for sensors [4], we assume that each sensor is aware of its own location.) This request includes a per-sensor reporting probability (PSRP) value p, such that each recipient sensor switches to an active state with probability p (without any inter-sensor interaction). An active sensor node generates a report at the appropriate time and transmits it back to the PSC. By iteratively adjusting the value of p broadcast over successive activation requests, the PSC directly controls the overall fraction of available nodes that become active, until the resulting number of active nodes satisfies the target N. Note that, in practice, the activation request may either be one-hop broadcast over the entire field by a high-power transmitter, or propagated to the appropriate nodes over a multi-hop infrastructure using techniques such as Directed Diffusion [5].

4.1. Overview of PSC Operation

To understand PSC's iterative operation, consider a sample query that requires the PSC to obtain 1000 samples (distinct readings) once every $T = 1$ min. from a region X over a duration $D = 1$ hour. Furthermore, assume that region X actually has 5000 sensors deployed, implying that the target bound is met for $p = 0.2$. To conserve energy, the PSC must permit $\approx$ 4000 nodes to turn their radios off and sleep for extended periods of time, while still retaining the ability to activate these nodes in case some of the initially chosen 1000 nodes fail or become otherwise unavailable. To satisfy both objectives, the PSC defines a macro-sampling frame (**MSF**), which consists of an initial probability adjustment phase (**IPAP**), followed by a non-adaptive data gathering phase (**DGP**). The sequence of PSC operations is illustrated in Figure 5. All the nodes in the region X keep their radios on to receive activation requests for the entire IPAP duration. During this time, the PSC broadcasts multiple activation requests, typically with varying p values, until the number of activated sensors satisfies the desired value. Since the PSRP computation is based on the number of reporting sensors (which report only once every T seconds), the IPAP duration is a time-varying multiple of T. At the end of the IPAP, the start of the DGP is explicitly broadcast by the PSC–at this point, the non-active nodes go to "sleep" for the specified duration of the DGP. There is clearly a tradeoff between the energy efficiency and responsiveness–a higher DGP duration increases the energy savings, but also implies a longer time till the PSC can adjust the PSRP again. Also, the PSC should converge to the correct p value fast, such that the IPAP interval remains small.

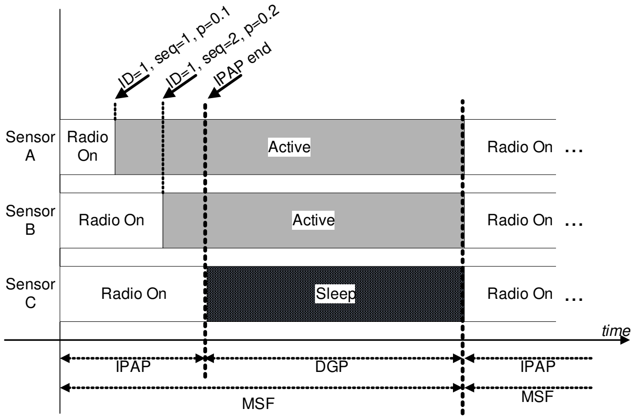


Figure 5. Timing diagram of IPAP and DGP phases in PSC

The important elements of each activation request (query) broadcast by the PSC are shown in Figure 6. Each query contains a unique *ID* number, and a monotonically increasing *sequence* number, where each sequence refers to a request with a newer PSRP value but the same ID. The request also contains a *type* field, which identifies the type of sensor nodes that are targeted. The request also contains a *samestream* field: if this field is set to 0, each new sequence number causes a sensor node to choose between the active and inactive state afresh. Conversely, if this field is set to 1, a currently active node continues to remain in the active state; only inactive nodes potentially switch to an active state. Having $samestream = 1$ thus allows the PSC to compose an incrementally richer set of active sources, while $samestream = 0$ allows a PSC to rapidly either increase or decrease the number of active nodes (chosen at random from the entire set of nodes within the specified region). Moreover, by using $samestream = 0$, and thus allowing nodes to recompute their active status at each new request, we can additionally statistically distribute the reporting burden over the entire reporting population (e.g., using techniques similar to LEACH[3]).

4.2. Controller Model for PSRP Computation

The key to the successful operation of the PSC is a mechanism for rapidly computing the correct p value, across a possibly wide variation in both N, the number of required sources, and S, the total number of available sensor

```
<sensor-query>
    <id>1179</id>
    <sequence>3</sequence>
    <type>thermal</type>
    <period>30</period>
    <samestream> 0</samestream>
    <psrp>0.05</psrp>
    <duration>3600</duration>
</sensor-query>
```

Figure 6. An example activation request

nodes. To achieve rapid convergence to the correct value without any direct knowledge of S, the PSC uses a PID (proportional-integral-derivative) controller for closed-loop feedback control of the PSRP. Figure 7 shows the PSRP controller's basic block diagram. The controller essentially continually *estimates* the total density of sensors ρ over the region from the obtained number of readings, and uses this to adjust the PSRP value. The PSRP is calculated to be proportional to the target number of reports, while inversely proportional to the density estimate.

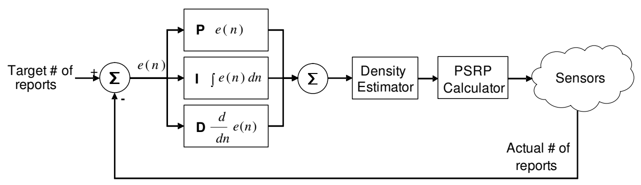
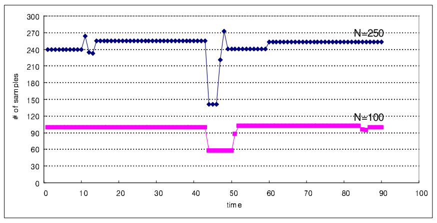


Figure 7. Block diagram of PSRP controller

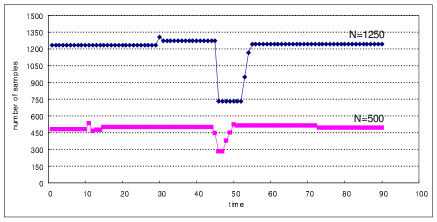
4.3. Initial Performance Evaluation of PSC

For the common case, when the number of sensors S remains constant across successive macro-signaling frames (MSF), the initial PSRP estimate remains valid and the IPAP (the phase where p is adjusted) consists of only 1 activation request. If the actual population of sensors changes, then the PID controller encounters a relatively small transient before it converges to the correct new value for ρ, and thus p. Note, however, that even in steady state, the number of received reports will vary over time from the ideal value N, due to the probabilistic activation model for each individual node. We, however, believe that this independent operational model is useful for sensor networks, which are often quite dense (large N), making the coefficient of variation acceptably small.

To demonstrate the performance of this closed-loop control, we have built a simulator for the interaction between the PSC and the available sensor nodes. Figure 8 shows a set of sample simulation results, where each graph represents the number of collected reports over $D = 90$ consecutive time slots (sampling period). In these graphs, the DGP (the interval when inactive nodes sleep) equals 10 time slots, and an IPAP (the iterative adjustment of p) terminates when the number of active sensors lies with $\pm 5\%$ of N. For each simulation run, to mimic an abrupt topological change, the number of available sensors was reduced to 60% of the initial value S at $T = 45$. We varied the total number of deployed sensors, S, as well as the target number of reports, N, desired by the PSC. Figure 8.a and 8.b show the results with the total number of sensors $S = 500$ and $S = 2500$ respectively. In each case, the target number of samples, N, was 20%, 50% of the total number of sensors. For simplicity, we assumed that the transmission of active sensor data to the PSC is reliable. The figures show that the PID controller rapidly converges to the correct p (maintains the desired level N) in all cases, and re-adjusts p with a very small transient (in the next MCF) even after a catastrophic loss of 40% of the sensor nodes. Moreover, as expected, the fluctuation in the actual number of active sensors is lower for larger values of S, and for larger values of the target N. In ongoing work, we are extending the PSC operation to adjust for packet losses within each sampling period T.



(a) S=500



(b) S=2500

Figure 8. Simulation results of the number of actually collected samples (N: target number of samples, S: Total number of deployed sensors)

5. Conclusion

MASTAQ is a data-management middleware that exposes a "statistical QoI" query API to sensor applications, and uses internal feedback control loops to adapt the volume of collected sensor data to meet these statistical bounds. These two properties make MASTAQ distinct from alternative sensor or stream-oriented middleware, such as Cougar [1], which conserves energy by distributing the query among sensor nodes, and SINA [9], which applies clustering-based in-network data aggregation techniques to reduce retransmission of similar information from geographically proximate sensors. Unlike [2], which also exposes a statistical API, MASTAQ does not aim to simply reduce energy consumption by learning correlation parameters from historical data. Instead, it tries to adjust the set of activated sensors dynamically when previously unseen events or disturbances occur. The use of the PSRP broadcast model, along with a closed loop controller, is a particularly attractive feature of MASTAQ, as it requires no coordination among sensor nodes.

In ongoing work, we are completing the development of the QSO component, after which we shall integrate the middleware with a sensor-network emulator to validate its functionality. Moreover, by using appropriate energy models, we shall quantify the energy savings for different levels of QoI preferences, and also quantify how statistical preferences affect the overall estimation/detection success rates. Finally, the closed-loop controller has to be extended to a) account for transmission losses from active nodes, and b) incorporate multi-hop routing overheads in the node selection algorithm.

References

[1] A. Demers, J. Gehrke, R. Rajaraman, N. Trigoni, and Y. Yao. The cougar project: A work-in-progress report. *ACM SIGMOD Record*, 32(4), 2003.

[2] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proceedings of VLDB*, 2004.

[3] W. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Trans. on Wireless Communications*, 1(4), 2002.

[4] J. Hightower and G. Boriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8), 2001.

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceeding of MOBICOM*, 2000.

[6] A. Kumar, P. Ishwar, and K. Ramchandran. On distributed sampling of smooth non-bandlimited fields. In *Proceedings of IPSN*, 2004.

[7] R. Nowark and U. Mitra. Boundary estimation in sensor networks: theory and methods. In *Proceedings of IPSN*, 2003.

[8] C. Olston and J. Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In *Proceedings of VLDB*, 2000.

[9] C. Srisathapornphat, C. Jaikaeo, and C. C. Shen. Sensor information networking architecture. In *Proceeding of International workshops on parallel processing*, 2000.