Processing Event-Monitoring Queries in Sensor Networks

Vassilis Stoumpos
University of Athens
Athens, Greece
stoumpos@di.uoa.gr

Antonios Deligiannakis
University of Athens
Athens, Greece
adeli@di.uoa.gr

Yannis Kotidis
Athens University of
Economics and Business
kotidis@aueb.gr

Alex Delis
University of Athens
Athens, Greece
ad@di.uoa.gr

Abstract

In this paper we present algorithms for building and maintaining efficient collection trees that provide the conduit to disseminate data required for processing monitoring queries in a wireless sensor network. While prior techniques base their operation on the assumption that the sensor nodes that collect data relevant to a specified query need to include their measurements in the query result at every query epoch, in many event monitoring applications such an assumption is not valid. We introduce and formalize the notion of event monitoring queries and demonstrate that they can capture a large class of monitoring applications. We then show techniques which, using a small set of intuitive statistics, can compute collection trees that minimize important resources such as the number of messages exchanged among the nodes or the overall energy consumption. Our experiments demonstrate that our techniques can organize the data collection process while utilizing significantly lower resources than prior approaches.

1 Introduction

Many pervasive applications rely on sensory devices that are able to observe their environment and perform simple computational tasks. Driven by constant advances in microelectronics and the economy of scale it is becoming increasingly clear that our future will incorporate a plethora of such sensing devices that will participate and help us in our daily activities. Even though each sensor node will be rather limited in terms of storage, processing and communication capabilities, they will be able to accomplish complex tasks through intelligent collaboration.

Nevertheless, building a viable sensory infrastructure cannot be achieved through mass production and deployment of such devices without addressing first the technical challenges of managing such networks. In this paper we focus in developing the necessary data collection infrastructure for supporting data-hungry applications that need to acquire and process readings from a large scale sensor network. While previous work has focused on optimizing specific types of queries such as aggregate [13], join [2], model-based [8, 12] and select-all [7, 19] queries, we propose a data dissemination framework that can address the needs of multiple, concurrent data acquisition requests in an efficient manner.

It is generally agreed that one cannot simply move the readings necessary for processing an application request out of the network and then perform the required processing in a designated node such as a *base station*. Wireless sensor nodes have limited energy capacity and such an approach will not only result in overburdening their radio links, but will also quickly drain their energy as radio transmission is by far the most important factor in energy consumption [14]. Thus, most recent proposals rely on building some type of ad-hoc interconnect for answering a query such as the *aggregation tree* [13, 24]. This is a paradigm of in-network processing that can be applied to non-aggregate queries as well [7]. In this paper we concentrate on building and maintaining efficient *data collection trees* that will provide the conduit to disseminate all data required for processing many concurrent queries in a sensor network, including long-term and ad-hoc type of queries, while minimizing important resources such as the number of messages exchanged among the nodes or the overall energy consumption.

While prior work [4, 20, 22] has also tackled similar problems, previous techniques base their operation on the assumption that the sensor nodes that collect data relevant to the specified query need to include their measurements (and, thus, perform transmissions) in the query result at every query *epoch*. However, in many monitoring applications such an assumption is not valid. Monitoring nodes are often interested in obtaining either the actual readings, or their aggregate values, from sensor nodes that detect interesting events. The detection of such events can often be identified by the readings of each sensor node. For example, in vehicle tracking and monitoring applications high noise levels may indicate the proximity of a vehicle. In military applications, high levels of detected chemicals can be used to warn nearby troops. In other applications, as in the case of approximate evaluation of queries over the sensor data [6, 15, 18], an event is defined when the current sensor reading deviates by more than a given threshold from the last transmitted value. In all of these scenarios, each sensor node is not forced to include its measurements in the query output at each epoch, but rather such a *query participation* is evaluated on a per epoch basis, depending on its readings and the definition

Aggregate Query		Non-Aggregate Query	
SELECT	AggrFun(s.value)	SELECT	s.id, s.value
FROM	Sensors s	FROM	Sensors s
WHERE	inclusionConditions(s) = true	WHERE	inclusionConditions(s) = true
SAMPLE	PERIOD e FOR t	SAMPLE	PERIOD e FOR t

Table 1: An Aggregate and a non-Aggregate Query over the Values Collected by the Sensor Nodes.

of interesting events. In this paper we term the monitoring queries where the participation of a node is based on the detection of an event of interest as *event monitoring queries* (EMQs).

Our techniques base their operation on collecting simple statistics during the operation of the sensor nodes. The collected statistics involve the number of events (or, equivalently, their frequency) that each sensor detected in the recent past. Our algorithms utilize these statistics as hints for the behavior of each sensor in the near future and periodically reorganize the collection tree in order to minimize certain metrics of interest, such as the overall number of transmissions or the overall energy consumption in the network. Our contributions are summarized as follows:

1. We introduce the notion of EMQs in sensor networks. EMQs are a superset of existing monitoring queries, but are handled uniformly in our framework, irrespectively of the minimization metric of interest.
2. We present detailed algorithms for minimizing important network resources such as the number of messages exchanged or the energy consumption during the execution of an EMQ. The presented algorithms are based on the collection and transmission of a small, and of constant size, set of statistics. We introduce our algorithms along with a succinct mathematical justification.
3. We extend our framework for the case of multiple concurrent EMQs of different types.
4. We present a detailed experimental evaluation of our algorithms. Our results demonstrate that our techniques can achieve a significant reduction in the number of transmitted messages, or the overall energy consumption, compared to alternative algorithms.

2 Motivational Example

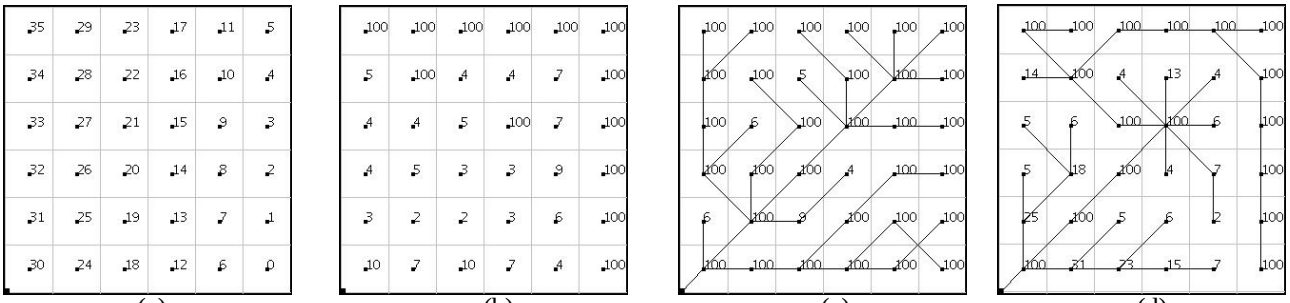
In Table 1 we present examples of the two main classes of monitoring queries in sensor networks. We borrow the syntax of TinyOS [13] to denote the epoch duration (e) and the lifetime of the query (t). The predicate *inclusionConditions* has been added in order to specify which sensor nodes will participate in the query evaluation per epoch. At each query epoch, all the sensor nodes that include their collected data in the query result are termed in our framework as *epoch participating nodes*. For queries that wish to collect data from all the sensor nodes at each epoch, the above predicate always evaluates to *true*.

When a monitoring query specifies inclusion predicates, these may contain either static or dynamic predicates (or both) regarding the sensor nodes. Examples of static predicates may involve, but are not limited to, the collection of measurements from: (i) Sensors with specific identifiers; (ii) Immobile sensors in a specific area; or (iii) Sensors monitoring a specific quantity, in cases of sensor networks with diverse types of sensor nodes that monitor different quantities. Static predicates are very useful in a variety of applications and have received the focus of the bulk of past research [13, 24].

However, there is a large class of monitoring queries that cannot be expressed using static inclusion conditions. Examples include vehicle tracking and equipment monitoring applications where inclusion predicates need to be conditioned on readings taken by the sensor nodes such as noise levels or temperature readings. In its most simple form a dynamic inclusion predicate may be a condition of the form "current reading $>$ threshold". More complex forms may require the evaluation of a user defined function over a history of accumulated readings. We call such predicates, whose evaluation depends also on the readings taken by the nodes, as dynamic predicates as they specify which nodes should include their response in the query evaluation at each epoch (i.e., nodes whose values exceed a given threshold, or deviate significantly from previous readings). We term those monitoring queries that contain dynamic predicates as *event monitoring queries* (EMQs). Given a monitoring query, existing techniques seek to develop *collection trees* that specify the way that the data is forwarded from the sensor nodes to the `Root` node. Periodically these collection trees may be reorganized in order to adapt to evolving data characteristics [18].

An important characteristic of EMQs, which is not taken into account by existing algorithms that design collection trees, is that each sensor node may participate in the query evaluation, by including its reading in the query result, only a limited number of times, based on how often the inclusion conditions are satisfied. We can thus associate an *epoch participation frequency* P_i with each sensor node S_i, which specifies the fraction of epochs that this node participated in the query result in the recent past.

Given estimates of the epoch participation frequencies, one can design significantly more efficient collection trees than prior approaches. Consider the sample scenario depicted in Figure 1(a). In this figure, 36 sensor nodes are placed in a grid. The sensor identifiers appear next to each sensor node. We also distinguish the `Root` node at the lower left corner, a monitoring node that performs queries over the data collected by the sensor nodes. In our sample network we assume that each sensor node can communicate with its immediate horizontal, vertical or diagonal neighbors, while only node S_{30} can communicate with the `Root` node. In Figure 1(b) we depict sample estimates for the number of times each sensor node will participate in the query result within the next 100 epochs. In the above scenario, given the presented epoch participation frequencies, two interior nodes along with all the boundary nodes on the upper and right-



(a) (b) (c) (d)

Figure 1: (a) Identifiers of sensors in grid arrangement; (b) Estimated number of participations in query result in 100 epochs; (c) Collection tree for MinHops algorithm. Cost = 3130 transmissions; (d) Collection tree for our algorithm. Cost = 1900 transmissions.

most edges of the network always detect events, while the remaining interior nodes detect events with a lower probability, whose average value is about 5%. For the aforementioned sample scenario, in Figure 1(c) we depict a sample collection tree chosen by an algorithm, termed as *MinHops* that seeks to minimize the number of hops that each node's data needs to traverse until it reaches the `Root` node. Next to each node we depict the actual number of transmissions that each node performed within these 100 epochs. Similarly, in Figure 1(d) we present the collection tree that our algorithms created for the evaluation of the SUM aggregate. A significant observation is that our algorithm seeks to forward the query results from nodes with high epoch participation frequencies through a limited number of interior nodes, compared to the *MinHops* algorithm. One can easily establish the significant reduction in the number of transmissions that our algorithm achieved (1900 vs 3130 or, equivalently, a 65% reduction).

3 Problem Formulation

We first introduce the types of EMQs that our framework supports and then present the optimization problems tackled in this paper. We then present the cost model used in our algorithms in order to estimate the energy consumption of a sensor node during the transmission process.

3.1 Supported Queries

In Table 1 we presented the two main classes of SQL queries that our framework supports. It is important to emphasize at this point that even non-participating nodes may take part in the query evaluation process by forwarding messages towards the `Root` node. However, the collected values of non-participating nodes influence neither the reported query result nor its size.

The first class of supported queries involve non-aggregate queries over the values of epoch participating sensor nodes. In this type of queries the amount of data transmitted by any node of the collection tree depends on the number of epoch-participating sensors that are descendants of that sensor node.

The second class of supported queries involves aggregate functions over the measurements collected by the participating sensor nodes. A good classification of aggregate functions is presented in [13], depending on the amount and type of state required in non-leaf nodes in order for them to calculate the aggregate result for the partition of descendant, in the collection tree, participating sensors. Table 2 summarizes this classification.

A crucial part of the operation of our algorithms is the estimation of the amount of data that will be transmitted in a given (or candidate) collection tree. In order to accurately estimate this information, the aggregate function being used needs to be distributive, algebraic or holistic (see Table 2). Unique and content-sensitive aggregate functions can only be supported by using a worst case estimate for the amount of transmitted data. Please note that holistic aggregate queries share similar characteristics with non-aggregate queries and, thus, are treated in a similar way in our framework.

3.2 Problem Definition

In this paper we seek to develop dissemination protocols for the classes of EMQs described in Section 3.1. The goal is, given the type of query at question, to design the collection tree so as to minimize either: (1) The number of transmitted messages in the network; or (2) The overall energy consumption in the network.

The minimization of additional metrics of interest is discussed in Section 4.4. Our algorithms do not make any assumptions about the placement of the sensor nodes, their characteristics or their radio models.

3.3 Energy Consumption Cost Model

A sensor node consumes energy at all stages of its operation. However, because our algorithms do not require any significant computational effort by the sensor nodes, we ignore in the cost model used in this paper the power consumption when the sensor node is idle and the consumption due to computations. The notation that will be used in our discussion here, and later in the description of our algorithms, is presented in Table 4. Additional definitions and explanations are presented in appropriate areas of the text.

We first describe the cost model used to estimate the energy consumption of a node S_i during the data transmission

Category	Type of Partial State Needed	State Size	Examples
Distributive	Aggregate values for descendants	Constant	MAX, MIN, COUNT, SUM
Algebraic	Aggregate values for descendants, but for different aggregate function	Constant	AVG
Holistic	Entire Data of descendants	Proportional to #epoch-participating descendants	MEDIAN
Unique	Distinct Values of descendants	Data-Dependent	COUNT DISTINCT
Content-Sensitive	Aggregate-Specific	Data-Dependent	Histogram of Values

Table 2. Characteristics and Examples of Aggregate Function Types.

Symbol	Typical Value
SC	$1\mu J$
E_{TX}	$50nJ/bit$
E_{RF}	$100pJ/bit/m^2$
E_{RX}	$50nJ/bit$

Table 3: Typical Radio Parameters.

Symbol	Description
`Root`	The node that initiates a query and which collects the relevant data of the sensor nodes
S_i	The i-th sensor node
P_i	The epoch participation frequency of S_i
D_i	The minimum distance, in number of hops, of S_i from the `Root`
$\|aggr\|$	The size of the (non-)aggregate values transmitted by a node
$E_{tr_{i,j}}$	Energy spent by S_i to transmit a new packet of $\|aggr\|$ bits to S_j
$DE_{tr_{i,j}}$	Energy spent by S_i to transmit additional $\|aggr\|$ bits to S_j (on an existing packet).
$AC_{i,j}$	Attachment cost of S_i to a candidate parent S_j
CF_i,DCF_i, HCF_i	Cost factors utilized by neighboring nodes of S_i when estimating their attachment cost to S_i

Table 4. Symbols Used in our Algorithm

of $|aggr| > 0$ bits of data to node S_j, which lies in distance $dist_{i,j}$ from S_i. The energy cost can be estimated using a linear model [17] as:

$$E_{tr_{i,j}} = SC_i + (H + |aggr|) \times (E_{TX_i} + E_{RF_i} \times dist_{i,j}^2),$$

where: (i) SC_i denotes the energy startup cost for the data transmission of S_i. This cost depends on the radio used by the sensor node; (ii) H denotes the size of the packet's header; (iii) E_{TX_i} denotes the per bit power dissipation of the transmitter electronics; and (iv) E_{RF_i} denotes the per bit and squared distance power delivered by the power amplifier. This power depends on the maximum desired communication range and, thus, from the distance of the nodes with which S_i desires to communicate. Thus, the additional energy consumption required to augment an existing packet from S_i to S_j with additional $|aggr|$ bits can be calculated as: $DE_{tr_{i,j}} = |aggr| \times (E_{TX_i} + E_{RF_i} \times dist_{i,j}^2)$.

For the case when each sensor node receives data, we need to keep in mind that each sensor must open its radio in order to receive data or queries transmitted by neighboring nodes. This startup cost is incurred when the node wakes up from its sleep mode and, in contrast to the data transmission case, is not directly related to the reception of data (since the sensor may receive no data). Thus, this mandatory cost is not taken into account in our model.

When a sensor node S_i receives $H + b_j$ bits from node S_j, then the energy consumed by S_i is given by: $E_{rec_i} = E_{RX_i} \times (H + b_j)$, where the value of E_{RX_i} depends on the radio model. Some typical values [17] of SC, E_{TX}, E_{RX} and E_{RF} are presented in Table 3.

4 Algorithm Overview

We now present our algorithms for creating and maintaining a collection tree that minimizes the desired metric (number of messages or energy consumption). We also provide detailed pseudocode in addition to a formal analysis.

4.1 Construction/Update of the Collection Tree

The algorithm is initiated with the query propagation phase. The query is propagated from the base station through the network using a flooding algorithm. In densely populated sensor networks, a node S_i may receive the announcement of the query from several of its neighbors. As in [13, 24] the node will select one of these nodes as its *parent node*. The chosen parent will be the one that exhibits the lowest *attachment cost*, meaning the lowest expected increase in the objective minimization function. For example, if our objective is to minimize the total number of transmitted messages, then the selection will be the node that is expected to result in the lowest increase in the number of transmitted messages in the *entire* path from that sensor until the `Root` node (and similarly for the rest of the minimization metrics). At this point we simply note that in order for other nodes to compute their attachment cost, node S_i transmits a small set of statistics $Stats_i$ and defer their exact definition for Section 4.2.

The result of this process is a collection tree towards the base station that initiated the flooding process. A key point in our framework is that the preliminary selection of a parent node may be revised in a second step where each node evaluates the cost of using one of its sibling nodes as an alternative parent. Due to the nature of the query propagation, and given simple synchronization protocols, such as those specified in [13], the nodes lying k hops from the `Root` node will receive the query announcement before the nodes that lie one hop further from the `Root` node. Let $RecS_k$ denote the set of nodes that receive the query announcement for the first time during the k-th step of the query propagation phase.

At step k of the query propagation phase, after the preliminary parent selection has been performed, each node S_i in set $RecS_k$, needs to consider whether it is preferable to alter its current selection and choose as its parent a *sibling node* within set $RecS_k - S_i$. Each node calculates a new set of statistics $Stats_i$, based on its preliminary parent selection, and transmits an *invitation*, which also includes the node's newly calculated $Stats_i$ values, that other nodes in $RecS_k$ (and only these nodes) may accept. Of course, we need to be careful at this point and make sure that at least one node within $RecS_k$ will not accept any invitation, as this would create a disconnected network and prevent nodes

from $RecS_k$ to forward their results to nodes belonging in $RecS_{k-1}$. We will achieve this by imposing a simple set of rules regarding when an invitation may be accepted by a sensor node.

Let $CandPar_i$ denote the set of nodes in $RecS_k$ that transmitted an invitation that S_i received. Let S_m be the preliminary parent node of S_i, as decided during query propagation. Amongst the nodes in $CandPar_i$, node S_i considers the node S_p such as the attachment cost $AC_{i,p}$ is minimized. If ties occur, then these are broken using the node identifiers (i.e., prefer the node with the highest id). Then S_p is selected as the parent of S_i *instead of the preliminary choice* S_m only if all of the following conditions apply:

- $AC_{i,p} < AC_{i,m}$. This conditions ensures that S_p seems as a better candidate parent than the current selection S_m.
- $AC_{i,p} \leq AC_{p,i}$. This conditions ensures that it is better to select S_p as the parent of S_i, than to select S_i as the parent of S_p.
- If $AC_{i,p} = AC_{p,i}$, then the identifier of S_p is also larger than the identifier of S_i. This condition is useful in order to allow nodes to forward messages through neighbor nodes in $RecS_k$ and also helps break ties amongst nodes and to prevent the creation of loops.

The collection tree may periodically get updated, either because of a significant change in data distribution or because of the addition/termination of queries in a multi-query setup discussed in section 5. Such updates are triggered by the base station using the same protocol used in the initial creation. In this case, the nodes compute and transmit their computed statistics in the same manner, but do not need to propagate the query itself.

4.2 Calculating the Attachment Cost

Determining the candidate parent with the lowest attachment cost is not an easy decision, as it depends on several parameters. For example, it is hard to quantify the resulting transmission probability of S_j, if a node S_i decides to select S_j as its parent node. In general, the transmission frequency of S_j (please note that this is different than the epoch participation frequency of the node) may end up being as high as $\min\{P_i + P_j, 1\}$ (when nodes transmit on different epochs) and as low as P_j (when transmissions happen on the same epochs and $P_i \leq P_j$). A commonly used technique that we have adopted in our work is to consider that the epoch participation by each node is determined by independent events. Using this independence assumption, node S_j will end up transmitting with a probability $P_i + P_j - P_i P_j$, an increase of $P_i(1-P_j)$ over P_j. Similarly, if S_{j-1} is the parent of S_j, this increase will also result in an increase in the transmission frequency of S_{j-1} by $P_i(1-P_j)(1-P_{j-1})$, etc. In our following discussion, for ease of presentation, when considering the attachment cost of S_i to a node S_j, we will assume that the nodes in the path from S_j to the `Root` node are the nodes $S_{j-1}, S_{j-2}, \ldots, S_1$.

4.2.1 Minimizing the Number of Transmissions

The attachment cost of S_i when selecting S_j as its parent node can be calculated by the increase in the transmission frequency of each link from S_i to the `Root` node as:

$$AC_{i,j} = P_i + P_i(1-P_j) + P_i(1-P_j)(1-P_{j-1}) + \ldots$$

A significant problem concerning the above estimation of $AC_{i,j}$ is that its value depends on the epoch participation frequencies of all the nodes in the path of S_j to the `Root` node. Since the number of these values depends on the actual distance, in number of hops, of S_j to the `Root` node, such a solution does not scale in large sensor networks.

Fortunately, there exists an alternative formula to calculate the above attachment cost. Our technique is based on a recursive calculation based on a single *cost factor* CF_i at each node S_i. In our example discussed above, the values of CF_i and $AC_{i,j}$ can be easily calculated as:

$$\begin{aligned} CF_i &= (1-P_i) \times (1+CF_j) \\ AC_{i,j} &= P_i \times (1+CF_j) \end{aligned}$$

One can verify that expanding the above recursive formula and setting as the boundary condition that the CF value of the `Root` node is zero gives the desired result. Thus, only the cost factor, which is a single statistic, is needed at each node S_j in order for all the other nodes to be able to estimate their attachment cost to S_j.

We also need to note that the formulas presented above also address the case of non-aggregate or holistic aggregate queries. In these cases the size of the transmitted data increases proportionally to the number of each node's epoch-participating descendants in the collection tree, as we approach the `Root` node. Thus, sometimes the transmitted data by a node may be split into multiple messages due to the maximum packet size. However, we first note that such cases typically occur in higher levels of the collection tree (and, thus, by a potentially small subset of the sensor nodes) and that, more importantly, our techniques seek to compute and utilize simple statistics. Our study of alternative cost models that incorporated this factor yielded only minor improvements while significantly increasing the communication cost during the collection tree formation. We thus omit such extensions from our presentation.

4.2.2 Minimizing Total Energy Consumption, Distributive and Algebraic Aggregates

This case is very similar to the case described above. When considering the attachment cost of S_i to a candidate parent S_j, we note that additional energy is consumed by nodes in the path of S_j to the `Root` node only if a new transmission takes place. This is because each node aggregates the partial results transmitted by its children nodes and transmits a new single partial aggregate for its sub-tree [13]. Thus, the size of the transmitted data is independent of the number of nodes in the subtree, only the frequency of transmission may get affected. Let $E_{tr_{i,j}}$ denote the energy consumption

when S_i transmits a message to S_j consisting of a header and the desired aggregate value(s) - based on whether this is a distributive or an algebraic aggregate function. The energy consumption follows the cost model presented in Section 3.3, where the P_{RF_i} value may depend on the distance between S_i and S_j (thus, the two indices used above). Using the above notation, and similarly to the previous discussion, the attachment cost $AC_{i,j}$ is calculated as:

$$\begin{aligned} AC_{i,j} &= P_i \times E_{tr_{i,j}} + P_i \times (1-P_j) \times E_{tr_{j,j-1}} + \\ &\quad P_i \times (1-P_j) \times (1-P_{j-1}) \times E_{tr_{j-1,j-2}} + \ldots \\ &= P_i \times (E_{tr_{i,j}} + CF_j), \qquad \textit{where} \\ CF_i &= (1-P_i) \times (E_{tr_{i,j}} + CF_j) \end{aligned}$$

If one wishes to take the receiving cost of messages into account, all that is required is to replace in the above formulas the symbols of the form $E_{tr_{k,p}}$ with $(E_{tr_{k,p}} + E_{rec_p})$, since each message transmitted by S_k to S_p will consume energy during its reception by S_p.

4.2.3 Minimizing Total Energy Consumption, Holistic Aggregate and Non-Aggregate Queries

When considering the attachment cost of S_i to a candidate parent S_j, we need not only consider the new messages generated in the path from S_j to the `Root` node, but also the energy consumption due to the increase in the length of messages that would have been transmitted anyway. Please recall that the energy consumption for each transmission of $|aggr|$ bits by S_i to S_j is given by: $DE_{tr_{i,j}} = |aggr| \times (P_{TX_i} + P_{RF_i} \times dist_{i,j}^2)$. Calculating the aforementioned number of messages is simple, as we have already discovered a similar recursive formula that estimates the attachment cost when only considering the transmission of new messages. So, we will utilize two new recursively computed statistics. The *DCF* value of a node will be similar to the *CF* value, but will use the $DE_{tr_{*,*}}$ transmission costs, instead of the $E_{tr_{*,*}}$ transmission costs used in the *CF* formula. The *HCF* value of a node will be equal to the sum of the $DE_{tr_{*,*}}$ values in the nodes path to the `Root` node. One can verify that the energy consumption due to the enlargement of messages, because of the attachment of S_i to S_j, that would have been transmitted anyway is: $P_i \times (HCF_j - DCF_j)$. The required formulas are presented below:

$$\begin{aligned} CF_i &= (1-P_i) \times (E_{tr_{i,j}} + CF_j) \\ HCF_i &= DE_{tr_{i,j}} + HCF_j \\ DCF_i &= (1-P_i) \times (DE_{tr_{i,j}} + DCF_j) \\ AC_{i,j} &= P_i \times (E_{tr_{i,j}} + CF_j) + P_i \times (HCF_j - DCF_j) \end{aligned}$$

4.2.4 Summary

Table 5 summarizes the statistics required to be transmitted by each node during the query propagation. Please note that the invitation phase always requires one more transmitted statistic, as the nodes need to check whether it is more beneficial to be attached to another node or the reverse (see the last two rules in Section 4.1) As it can be clearly seen from this table, our algorithms utilize only a limited number of statistics, which are computed using only information transmitted by neighboring sensor nodes. Due to space constraints, the proof of the following Theorem can be found in the full paper [21].

Minimization Metric	Type of Aggregate	Decision	Invitation
Transmissions	Aggregate Non-Aggregate	CF_i	P_i, CF_i
Energy Consumption	Distributive Algebraic	CF_i	P_i, CF_i
Energy Consumption	Holistic Non-Aggregate	CF_i, HCF_i, DCF_i	P_i, CF_i, HCF_i, DCF_i

Table 5. Statistics Attached to Messages

Theorem 1 *For sensor networks that satisfy the connectivity requirements of Section 3 our algorithm always creates a connected routing path that avoids loops.*

4.3 Algorithm Implementation

In Algorithm 1 we present the complete algorithm for the decisions of a sensor node. This algorithm is invoked both at the query propagation phase and when updating the collection tree. Each node first waits to receive the decisions by nodes that lie one hop closer to the `Root` node (Line 2). Based on the received decisions it performs an initial parent selection using the *ProcessDecisions* subroutine described in Algorithm 2 (Lines 3-4). It then calculates some necessary statistics and transmits an invitation to neighboring nodes (Lines 5-6). The node then waits (Line 7) to receive invitations from neighboring nodes and makes a final decision on its parent selection using the *ProcessInvitations* subroutine presented in Algorithm 3 (Lines 8-9). The node then transmits its final decision (Line 10) to neighboring nodes and ignores any received decisions or invitations until the next update period when the collection tree will be reorganized (a counter denoting the reorganization period can be attached to the queries transmitted by the `Root` node in order to help the nodes understand the transition to a new update period). An interesting observation that we have not mentioned so far involves the nodes with zero epoch participation frequencies. For these nodes, the computed attachment costs to any neighboring node will also be zero. In such cases we select the candidate parent which produces the lowest $E_{tr_{i,j}} + CF_j + HCF_j - DCF_j$ value. This decision is expected to minimize the attachment cost, if the sensor at some point starts observing events.

4.4 Extensions

In the full paper [21] we describe extensions on refining the statistics utilized by the sensor nodes. Furthermore, we show that our techniques can be easily adapted to incorporate different minimization metrics, than the ones presented in Section 3.2. For example, the formulas for minimizing the number of transmitted bits can be derived using the formulas for the energy minimization for the corresponding

Algorithm 1 `BuildCollectionTree()` Subroutine

1: {S_i is the node being examined}
2: Wait to receive decisions by neighboring nodes
3: Set $\overrightarrow{Dec}$ as the received decisions by the nodes with minimum D values (ignore other decisions).
4: k = ProcessDecisions($\overrightarrow{Dec}$) {Returns index of selected parent}
5: $D_i = 1 + D_k$
6: Transmit invitation to neighboring nodes
7: Wait to receive invitations by neighboring nodes
8: Set $\overrightarrow{Inv}$ as the received invitations by the nodes with D values equal to D_i (ignore other invitations).
9: m = ProcessInvitations($\overrightarrow{Inv}$) {Returns index of selected parent}
10: Transmit decision
11: Ignore received decisions and invitations until next reorganization.

Algorithm 2 `ProcessDecisions(`$\overrightarrow{Dec}$`)` Subroutine

1: {S_i is the node being examined}
2: Select Dec_k as the decision with the minimum attachment cost. If $P_i = 0$ utilize in the calculations a non-zero value at this step to prevent all nodes from having the same (zero) attachment cost.
3: Let S_k be the sender of Dec_k
4: Set $parent(S_i) = S_k$
5: Calculate statistics (cost factors) for current node based on current parent selection
6: Return k {Index of selected parent node}

type of query (i.e., distributive, non-aggregate). In these formulas one simply has to substitute the term $E_{tr_{i,j}}$ with the size of a packet (including the packet's header) and to substitute the term $DE_{tr_{i,j}}$ with the size of each transmitted aggregate value (thus, ignoring the header size). In the case where the goal is to maximize the minimum energy amongst the sensor nodes, the attachment cost can be derived from the minimum energy, amongst the nodes in a sensor's path to the `Root` node, raised to -1 (since our algorithms select the candidate parent with the *minimum* attachment cost).

5 Multi-Query Optimization

In the multi-query scenario, each node S_i may choose different parent nodes for each posed query. Thus, the resulting network topology may not be a tree after-all but a directed acyclic graph. In the case of multiple concurrent queries we need to introduce some additional (or augmented) notation for the presentation of our algorithms. Let P_i^k denote the epoch participation frequency of S_i regarding the k-th query. Let $f_i(k)$ denote the index of the selected parent node of S_i for the k-th posed query.

In order to be able to derive recursive formulas for the estimation of the attachment cost, in our approach we break the posed queries into two groups. The first group of queries contains the distributive and algebraic aggregate queries, while the second group contains the holistic and non-aggregate queries. In our discussion below we assume that the group of queries handled in each case contains a total of M queries of similar type.

Using the notation $PROD_i^k = \prod_{\substack{x=1 \\ f(x)=f(k)}}^{M} (1-P_i^x)$ and $partialPROD_{i,j}^k = \prod_{\substack{x<k \\ f(x)=S_j}} (1-P_i^x)$, and by processing the

Algorithm 3 `ProcessInvitations(`$\overrightarrow{Inv}$`, k)` Subroutine

1: {S_i is the node being examined}
2: {S_k is the current parent node}
3: In the following discussion, all estimations of the attachment cost utilize the same E_{RF_i} value, as discussed at the end of Section 4.2.3.
4: Select Inv_m as the invitation with the minimum attachment cost. If $P_i = 0$ utilize in the calculations a non-zero value at this step to prevent all nodes from having the same (zero) attachment cost.
5: Let S_m be the sender of Inv_m
6: **if** $AC_{i,m} \leq AC_{i,k}$ **then**
7: Return k {No benefit in changing parent node}
8: **end if**
9: Calculate $AC_{m,i}$ using information from Inv_m
10: **if** $AC_{i,m} > AC_{m,i}$ **then**
11: Return k {Reverse decision is better}
12: **else if** $AC_{i,m} == AC_{m,i}$ AND $i > m$ **then**
13: Return k {Base decision on identifier}
14: **end if**
15: Set $parent(S_i) = S_m$
16: Calculate statistics (cost factors) for current node based on current parent selection
17: Return m {Index of selected parent node}

queries in order based on their identifier (i.e., from 1 to M), we demonstrate in the full paper [21] that the attachment cost $AC_{i,j}^k$ of S_i to S_j regarding the k-th query is calculated as follows:

- Minimizing Total Number of Transmissions.

$$\begin{aligned} AC_{i,j}^k &= P_i^k \times (JCF_j^k + partialProd_{i,j}^k) \\ JCF_i^k &= PROD_i^k + (1-P_i^k) \times JCF_j^k \end{aligned}$$

- Minimizing Total Energy Consumption: Distributive and Algebraic Aggregates.

$$\begin{aligned} AC_{i,j}^k = & P_i^k \times partialProd_{i,j}^k \times E_{tr_{i,j}} + \\ & P_i^k \times (1-partialProd_{i,j}^k) \times DE_{tr_{i,j}} \\ & P_i^k \times JCF_j^k + P_i^k \times (JECF_j^k - JDCF_j^k) \\ JCF_i^k = & \ PROD_i^k \times E_{tr_{i,f(k)}} + (1-P_i^k) \times JCF_{f(k)}^k \\ JECF_i^k = & \ (1-P_i^k) \times (DE_{tr_{i,f(k)}} + JECF_{f(k)}^k) \\ JDCF_i^k = & \ PROD_i^k \times DE_{tr_{i,f(k)}} + (1-P_i^k) \times JDCF_{f(k)}^k \end{aligned}$$

- Minimizing Total Energy Consumption: Holistic Aggregate and Non-Aggregate Queries.

$$\begin{aligned} AC_{i,j}^k = & P_i^k \times partialProd_{i,j}^k \times E_{tr_{i,j}} + \\ & P_i^k \times (1-partialProd_{i,j}^k) \times DE_{tr_{i,j}} + \\ & P_i^k \times JCF_j^k + P_i^k \times (JHCF_j^k - JDCF_j^k) \\ JCF_i^k = & PROD_i^k \times E_{tr_{i,f(k)}} + (1-P_i^k) \times JCF_{f(k)}^k \\ JHCF_i^k = & DE_{tr_{i,f(k)}} + JHCF_{f(k)}^k \\ JDCF_i^k = & PROD_i^k \times DE_{tr_{i,f(k)}} + (1-P_i^k) \times JDCF_{f(k)}^k \end{aligned}$$

6 Experiments

We developed a simulator for testing the algorithms proposed in this paper under various conditions. In our discussion we term our algorithm for minimizing the number

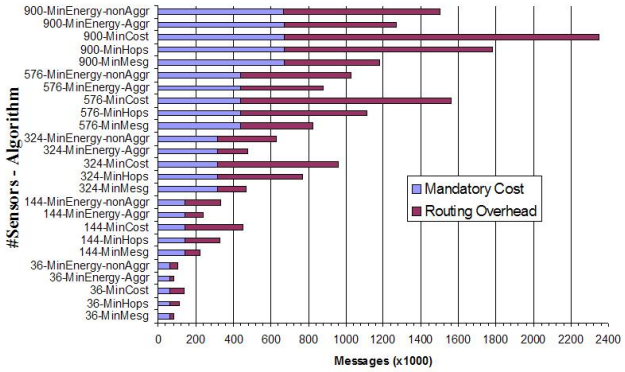


Figure 2: Messages and Message Overhead for Synthetic Data Set. Results for MinEnergy Presented for both Aggregate SUM and Non-Aggregate Query.

of transmissions as *MinMesg*, and our algorithm for minimizing the overall energy consumption as *MinEnergy*. Our techniques are compared against two intuitive algorithms. In the *MinHops* algorithm, each sensor node that receives the query announcement randomly selects as its parent node a sensor amongst those with the minimum distance, in number of hops, from the `Root` node [13]. In the *MinCost* algorithm, each sensor seeks to minimize the sum of the squared distances amongst the sensors in its path to the `Root` node, when selecting its parent node. Since the energy consumed by the power amplifier in many radio models depends on the square of the communication range, the *MinCost* algorithm aims at selecting paths with low communication cost.

In all sets of experiments we place the sensor nodes on random locations over a rectangular area. The radio parameters were set accordingly to the values in Table 3. The message header was set to 32 bits, similarly to the size of each statistic and aggregate value. In all figures we account for the overhead of transmitting statistics and invitation messages during the creation of the collection tree in our algorithms.

6.1 Experiments with Synthetic Data Sets

We initially placed 36 sensor nodes in a 300x300 area, and then scaled up to the point of having 900 sensors. We set the maximum broadcast range of each sensor to 90m. In all cases the `Root` node was placed on the lower left part of the sensor field. In each case we set the epoch participation frequency of the sensor nodes with the maximum distance, in hop count, from the `Root` to 1. Unless specified otherwise, with probability 8% some interior node assumed an epoch participation frequency of 1, while the epoch participation frequency of the remaining interior nodes was set to 5%.

We first evaluated a non-aggregate "SELECT *" query over the measurements obtained by the epoch participating sensor nodes. Since in this query the measurements of each epoch participating node need to be propagated all the way to the `Root` node, and the only sharing that can be achieved in combined messages involves the message's header, we expect little energy savings in this case. We also evaluated a SUM aggregate query over the values of epoch participating sensor nodes using all algorithms. We present the total number of transmissions for each type of query, algorithm and number of sensors in Figure 2. Please note that the MinEnergy algorithm built very different collection trees for the two types of queries. For the remaining algorithms, the number of transmitted messages was the same for both types of queries. The corresponding average energy consumption by the sensor nodes for each case is presented in Table 6.

As we can see, our *MinMesg* algorithm achieves a significant reduction in the number of transmitted messages compared to the MinHops and MinCost algorithms. The reduction in messages is up to 64% and 105%, respectively, with an average gain of 48% and 93.7%, respectively, compared to the the *MinHops* and *MinCost* algorithms. However, since these gains depend on the number of transmissions that epoch-participating nodes perform, it is perhaps more interesting to measure the *routing overhead* of each technique. We define the routing overhead of each algorithm as the relative increase in the number of transmissions when compared to the number of epoch participations by the sensor nodes. Note that the latter number is a *mandatory* cost that represents the transmissions in the network if each sensor could communicate directly with the `Root` node. For example, if the total number of epoch participations by the sensor nodes was 1000, but the overall number of transmissions was 1700, then the routing overhead would have been equal to $(1700-1000)/1000 = 70\%$. As we observe from Figure 2, our *MinMesg* algorithm often results in 3 times smaller routing overhead compared to the alternative algorithms considered. We also observe that the MinEnergy algorithm in the aggregate case produced results very close to the ones of MinMesg. A main difference between these two algorithms is that amongst candidate parents with similar cost factors, the MinEnergy algorithm is less likely to select a distant neighbor than the MinMesg algorithm, which only considers epoch participation frequencies. This is a trend that we observed in all our experiments. However, in the case of non-aggregate queries, the MinEnergy algorithm formed very different collection trees, as it avoided routing measurements through very long paths.

The MinEnergy algorithm performs very well in both types of queries. Compared to the MinHops algorithm, it achieves up to a 2-fold reduction in the power drain for aggregate queries and up to 19% for non-aggregate queries. Compared to the MinCost algorithm the energy savings are smaller but still significant (i.e., up to 79% in the aggregate query). The MinMesg algorithm is obviously a very poor choice, with respect to the energy consumption, for non-aggregate queries.

We expect that the more the epoch participation frequencies of sensor nodes increase, the less likely that out tech-

Sensors	Aggregate SUM Query				Non-Aggregate "SELECT *" Query			
	MinMesg	MinEnergy	MinHops	MinCost	MinMesg	MinEnergy	MinHops	MinCost
36	109.339	109.341	161.278	136.354	381.483	292.231	335.920	303.270
144	70.129	68.971	139.821	121.640	515.215	344.489	390.806	344.213
324	71.662	68.703	146.425	106.416	687.083	444.157	523.670	448.816
576	65.921	64.717	127.315	104.156	624.817	457.788	547.147	471.845
900	67.107	64.077	128.299	102.708	756.902	549.262	640.830	559.183

Table 6: Average Power Consumption (in mJ) for Synthetic Dataset

# Sensors	MinMesg	MinEnergy	MinHops	MinCost
150	73.607	67.821	111.751	91.990
600	58.131	58.273	97.958	85.158
1350	50.418	49.350	89.231	76.099

Table 7: Average Power Consumption (in mJ) for SchoolBuses Dataset

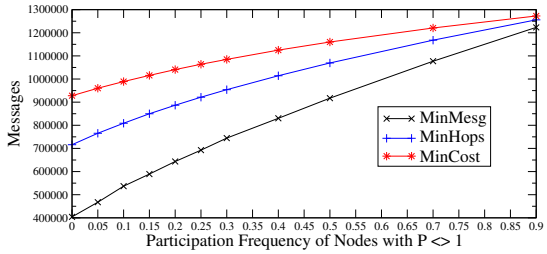


Figure 3: Transmissions Varying the Epoch Participation Frequency

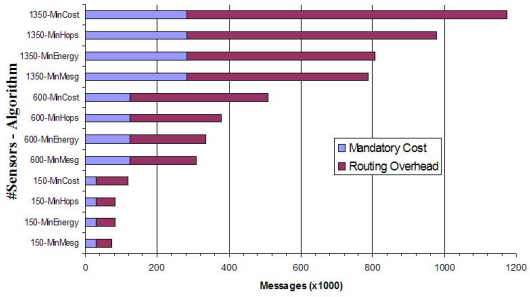


Figure 4. Transmissions - Trucks data

# Sensors	Transmissions		Average Energy Consumption	
	MinEnergy-1Step	MinEnergy	MinEnergy-1Step	MinEnergy
36	106474	79887	147.882	109.341
144	405279	232298	126.374	68.971
324	749013	487889	112.964	68.703
576	1258888	827959	101.935	64.717
900	1812024	1279129	93.273	64.077

Table 8: Comparison of 1-Step and 2-Step Parent Selection for MinEnergy Algorithm. Number of Transmissions and Average Power Consumption (in mJ) for Synthetic Data Set

niques will be able to provide substantial savings compared to the MinHops and MinCost algorithms. In Figure 3 we repeat the aggregate query of Figure 2 at the sensor network with 324 nodes, but vary the epoch participation frequency P_i of those nodes that do not make a transmission at each epoch (i.e., of those nodes with $P_i < 1$). While Figure 3 validates our intuition, it also demonstrates that significant savings can be achieved even when sensor nodes have large P_i values (i.e., $P_i \geq 0.5$).

A novel feature of our technique is the 2-step parent selection phase. In Table 8 we compare the performance of our MinEnergy algorithm in the aggregate SUM query described above versus a variant that was not allowed to select

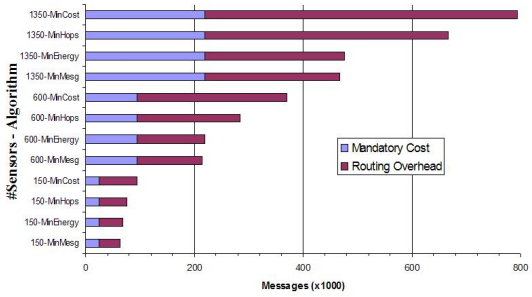


Figure 5. Transmissions - SchoolBuses data

a node's sibling as its parent node in the collection tree. As we can see, the benefits from utilizing the 2-step process are important in all aspects (transmitted messages and power consumption).

6.2 Experiments with Real Data Sets

We also experimented with the following two real data sets. The **Trucks** data set contains trajectories of 276 moving trucks [1]. Similarly, the **SchoolBuses** data set contains trajectories of 145 moving schoolbuses [1]. For each data set we initially overlaid a sensor network of 150 nodes over the monitored area. We set the broadcast range such that interior sensor nodes could communicate with at least 5 more sensor nodes. Moreover, each sensor could detect objects within a circle centered at the node and with radius equal to 60% of the broadcast range. We then scaled the data set up to a network of 1350 sensors, while keeping the sensing range steady. In Figures 4 and 5 we depict the total number of transmissions by all algorithms for the Trucks and SchoolBuses data sets, correspondingly, when computing the SUM of the number of detected objects. In our scenario, nodes that do not observe an event make a transmission only if they need to propagate measurements/aggregates by descendant nodes. Due to space constraints we present the average energy consumption of the sensor nodes in the same experiment for only the SchoolBuses data set in Table 7. As it is evident, our algorithms achieve significant savings in both metrics. For example, the MinCost algorithm, which exhibits lower power consumption than the MinHops algorithm, still drains about 50% more energy than our MinEnergy algorithm. Moreover, both our MinMesg and MinEnergy algorithms significantly reduce the amount of transmitted messages by up to 42% and 73% when compared to the MinHops and MinCost algorithms, respectively.

We then decided to mix the data sets. We separated

# Sensors	MinMesg	MinEnergy	MinHops	MinCost
150	70,583	125,686	107,887	122,532
600	286,894	401,431	336,143	479,544
1350	430,094	774,530	606,105	1,038,093

Table 9. Messages for Multi-Query Scenario

the SchoolBuses into two categories (by randomly coloring each schoolbus as either color *A* and *B*) and overlaid this data set with the trucks data set. We then performed three simultaneous queries requesting the total number of trucks, schoolbuses of color *A* and schoolbuses of color *B* observed in the network. We used the same topology, network scale and placement of the `Root` node as above and compared our *MinMesg* and *MinEnergy* algorithm with the *MinHops* and *MinCost* algorithms, which were modified to select a single parent node for all queries (this produced the best results for them). Due to space constraints, in Table 9 we depict only the total number of transmitted messages for all algorithms.

7 Related Work

The database community has long been the advocate of using an embedded database management system for data acquisition in sensor networks [13, 24]. The use of a declarative SQL-like query interface allows rapid development of applications in such systems without the need to manage hand-coded programs at each sensor node [14]. In the database community different types of popular queries have been discussed, such as aggregate [5, 6, 13, 18, 16], join [2], model-based [8, 12] and select-all queries [7, 19]. Tracking queries that seek to determine the spatial extent of a particular phenomenon have also been considered [9, 23].

Many of the low-level networking details have already been discussed in the networking community and, thus, can be utilized in our framework. As an example, nodes in unattended wireless networks must be able to self-configure [3] and discover their surrounding nodes [10]. Prior work on computing energy-efficient data routing paths (such as the aggregation tree) [11, 20, 22] have tackled similar problems, but these techniques base their operation on the assumption that the sensor nodes that collect data relevant to the specified query need to include their measurements in the query result at every query epoch. However, this assumption does not hold in event monitoring queries that are the scope of our framework. Due to space constraints, a more elaborate discussion of the related work can be found in [21].

8 Conclusions

In this paper we presented algorithms for building and maintaining efficient collection trees in support of event monitoring queries in wireless sensor networks. We demonstrated that is it possible to create efficient collection trees that minimize important network resources using a small set of statistics that are communicated in a localized manner during the construction of the tree topology. Furthermore, our techniques utilize a novel 2-step refinement process that significantly increases the quality of the created trees. We have also demonstrated that our algorithms can handle a mix of event monitoring queries (EMQs) including aggregate and non-aggregate queries.

References

[1] Rtree Pportal. http://www.rtreeportal.org.

[2] D.J. Abadi, S. Madden, and W. Lindenr. REED: Robust, Efficient Filtering and Event Detection in Sensor Networks. In *VLDB*, 2005.

[3] A. Cerpa and D. Estrin. ASCENT: Adaptive Self-Configuring sEnsor Network Topologies. In *INFOCOM*, 2002.

[4] Jae-Hwan Chang and Leandros Tassiulas. Energy Conserving Routing in Wireless Ad-hoc Networks. In *INFOCOM*, 2000.

[5] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate Aggregation Techniques for Sensor Databases. In *ICDE*, 2004.

[6] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Hierarchical In-Network Data Aggregation with Quality Guarantees. In *EDBT*, 2004.

[7] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Dissemination of Compressed Historical Information in Sensor Networks. *VLDB Journal*, 2007.

[8] A. Deshpande, C. Guestrin, S. Madden, J.M. Hellerstein, and W. Hong. Model-Driven Data Acquisition in Sensor Networks. In *VLDB*, 2004.

[9] M. Duckham, S. Nittel, and M. Worboys. Monitoring Dynamic Spatial Fields Using Responsive Geosensor Networks. In *GIS*, 2005.

[10] D. Estrin, R. Govindan, J. Heidermann, and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *MobiCOM*, 1999.

[11] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidermann. Impact of Network Density on Data Aggregation in Wireless Sensor Networks. In *ICDCS*, 2002.

[12] Y. Kotidis. Snapshot Queries: Towards Data-Centric Sensor Networks. In *ICDE*, 2005.

[13] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A Tiny Aggregation Service for ad hoc Sensor Networks. In *OSDI Conf.*, 2002.

[14] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The Design of an Acquisitional Query processor for Sensor Networks. In *ACM SIGMOD*, 2003.

[15] C. Olston and J. Widom. Offering a Precision-Performance Tradeoff for Aggregation Queries over Replicated Data. In *VLDB*, 2000.

[16] S. Pattem, B. Krishnamachari, and R. Govindan. The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks. In *IPSN*, 2004.

[17] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2), 2002.

[18] A. Sharaf, J. Beaver, A. Labrinidis, and P. Chrysanthis. Balancing Energy Efficiency and Quality of Aggregate Data in Sensor Networks. *VLDB Journal*, 2004.

[19] A. Silberstein, R. Braynard, and J. Yang. Constraint Chaining: On EnergyEfficient Continuous Monitoring in Sensor Networks. In *SIGMOD*, 2006.

[20] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *ACM/IEEE International Conference on Mobile Computing and Networking*, 1998.

[21] V. Stoumpos, A. Deligiannakis, Y. Kotidis, and A. Delis. Processing Event-Monitoring Queries in Sensor Netwrks. Technical Report, University of Athens, June 2007. Available at http://www.cs.umd.edu/ adeli/TR07.pdf.

[22] N. Trigoni, Y. Yao, A.J. Demers, J. Gehrke, and R. Rajaraman. Multi-query Optimization for Sensor Networks. In *DCOSS*, 2005.

[23] W. Xue, Q. Luo, L. Chen, and Y. Liu. Contour Map Matching for Event Detection in Sensor Networks. In *SIGMOD*, 2006.

[24] Y. Yao and J. Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. *SIGMOD Record*, 31(3):9–18, 2002.