Building Efficient Aggregation Trees for Sensor Networks' Event-Monitoring Queries

Antonios Deligiannakis[1], Yannis Kotidis[2], Vassilis Stoumpos[3], and Alex Delis[3]

[1] Technical University of Crete
[2] Athens University of Economics and Business
[3] University of Athens

Abstract. In this paper we present algorithms for building and maintaining efficient aggregation 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 aggregation 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 aggregation 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 on developing the necessary data aggregation infrastructure for supporting aggregate queries. For such applications, most recent proposals rely on building some type of ad-hoc interconnect for answering a query such as the *aggregation tree* [16, 26]. 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 *aggregation trees* that will provide the conduit to disseminate all data required for processing aggregate queries, while minimizing important resources such as the number of messages exchanged among the nodes or the overall energy consumption.

While prior work [4, 23, 24] 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 aggregate values only from sensor nodes that detect interesting events. In such applications, each sensor node is not forced to include its measurements in the aggregate at each epoch, but rather such a *query participation* is evaluated on a per epoch basis, depending on its readings and the definition 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 aggregation tree in order to minimize certain metrics of interest, such as the overall number of transmissions or the overall energy consumption in the network. The formation of the aggregation tree is based on the aggregation and local transmission of only a small set of values at each node termed as *cost factors* in our framework. Using these cost factors each sensor selects its parent node, through which it will forward its results towards the base station, based on the estimated corresponding *attachment cost*. In a nutshell, the attachment cost of a parent selection is the increase in the objective function (i.e., the number of transmitted messages) resulting from this selection. Given the estimates of attachment costs that our algorithms compute, our work demonstrates that they are able to design significantly better aggregation trees than existing techniques.
Our contributions are summarized as follows:

1. We formally 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 metrics such as the number of messages exchanged or the energy consumption during the execution of an aggregate EMQ. The presented algorithms are based on the aggregation and transmission of a small, and of constant size, set of statistics. We introduce our algorithms along with a succinct mathematical justification.
3. 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 Related Work

The database community has long been the advocate of using an embedded database management system for data acquisition in sensor networks [16, 26]. The use of a

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

Table 1. An Aggregate Query over the Values Collected by Sensor Nodes.

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 [17].

In the database community different types of popular queries have been discussed, such as aggregate [5, 6, 16, 21, 19], join [2], model-based [9, 14] and select-all queries [7, 22]. Tracking queries that seek to determine the spatial extent of a particular phenomenon have also been considered [10, 25]. In [16] the nodes are first organized in a tree topology, termed the aggregation tree. During query execution, each epoch is subdivided into intervals and parent nodes in the aggregation tree listen for messages containing partial aggregates from their children nodes during pre-defined time-slots. Another notable method for synchronizing the transmission periods of nodes is the recently proposed wave scheduling approach of [8]. The work in [28] describes a framework that profiles recent data acquisition activity by the nodes and computes their waking window though an in-network execution of the critical path method. This technique is complementary to our work, as they help identify a proper scheduling for data transmission by the nodes, while our methods focus on optimizing the routing topology.

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 [11]. Prior work on computing energy-efficient data routing paths (such as the aggregation tree) [13, 23, 24] 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. In the other end of the spectrum, the work in [15] and [12] discuss join and aggregation queries involving rare events. Thus, they follow an alternative path, which is to construct the data collection network on-the-fly when such events occur. However, this practice in unsuitable for out setting involving sensor nodes with both low and high participation frequencies, since it would incur a high overhead for frequently maintaining the collection network. Furthermore, the work in [12] assumes the existence of a high speed connection for all nodes at the boundaries of the network, through which the data that reaches the boundary nodes can be communicated.

3 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 [16] 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 [16, 26]. Inclusion conditions that contain only static predicates result in a fixed subset of the sensor nodes participating in the query output at each epoch. This allows for simple data dissemination and collection protocols based on fixed aggregation trees that need to be altered only when either node or communication failures exist.

However, there exists 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. In the case of approximate evaluation of queries over the sensor data [6, 18, 21], the inclusion predicate is satisfied when the current sensor reading deviates by more than a given threshold from the last transmitted value. 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 *aggregation trees* that specify the way that the data is forwarded from the sensor nodes to the `Root` node. Periodically these aggregation trees may be reorganized in order to adapt to evolving data characteristics [21].

An important characteristic of EMQs, which is not taken into account by existing algorithms that design aggregation 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.

4 Problem Formulation

Our current framework supports distributive (i.e., COUNT, SUM, MAX, MIN) and algebraic (i.e., SUM) queries involves aggregate functions over the measurements col-

lected by the participating sensor nodes. A good classification of aggregate functions is presented in [16], 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 aggregation tree, participating sensors. In our future work we plan to extend our framework to support all types of aggregate and non-aggregate queries.

4.1 Problem Definition

In this paper we seek to develop dissemination protocols for distributive and algebraic EMQs. The goal is, given the type of query at question, to design the aggregation tree so as to minimize either:

1. The number of transmitted messages in the network.
2. The overall energy consumption in the network.

Our algorithms do not make any assumptions about the placement of the sensor nodes, their characteristics or their radio models. However, in order to simplify the presentation, in our discussion we will focus on networks where any communication between pairs of sensor nodes is either bidirectional or impossible.

4.2 Energy Consumption Cost Model

A sensor node consumes energy at all stages of its operation. However, this energy consumption is minimal when the sensor is in a sleep mode. Furthermore, the energy drain due to computations may, in some applications, be significant, but it is typically much smaller than the cost of communication [17]. Due to this fact and because our algorithms do not require any significant computational effort by the sensor nodes, we ignore in the cost model the power consumption when the sensor node is idle and the consumption due to computations. We will thus focus on capturing the energy drain due to data communication in data driven applications. More particular, we need to estimate the energy consumption of a node S_i when either transmitting, receiving or idle listening for data. The notation that will be used in our discussion here, and later in the description of our algorithms, is presented in Table 3. 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 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 [20] 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: $DE_{tr_{i,j}} = |aggr| \times (E_{TX_i} + E_{RF_i} \times dist_{i,j}^2)$.

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

Table 2. 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 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	Cost factor utilized by neighboring nodes of S_i when estimating their attachment cost to S_i

Table 3. Symbols Used in our Algorithm

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 [20] of SC, E_{TX}, E_{RX} and E_{RF} are presented in Table 2.

The energy consumed by a sensor node when idle listening for data is significant and often comparable to the energy of receiving data. For example, in the popular MICA2 nodes the ratios for radio power draw during idle-listening, receiving of a message and transmission are 1:1:1.41 at 433MHz with RF signal power of 1mW in transmission mode [27]. Thus, due to the similar energy consumption by a sensor while either receiving or idle listening for data, our algorithms focus on the energy drain during the transmission of data.

5 Algorithm Overview

We now present our algorithms for creating and maintaining an aggregation tree that minimizes the desired metric (number of messages or energy consumption) for algebraic or distributive aggregate EMQs. Our algorithms are based on a top-down formation of the aggregation tree. The intuition behind such an approach is that the epoch participation frequency of each node in the aggregation tree influences the transmission frequency of only nodes that lie in its path to the `Root`. We thus demonstrate in this section that estimating the magnitude of this influence can be easily achieved by a top-down construction of the aggregation tree, while requiring the transmission of only a small set of statistics.

5.1 Construction/Update of the Aggregation Tree

The algorithm is initiated with the query propagation phase and periodically, when the aggregation tree is scheduled for reorganization. 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 [16, 26] 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 5.2.

The result of this process is an aggregation 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 [16], 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 $RecS_k - S_i$.[4] 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).[5] 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.

[4] Please note that at this step any initially selected parent of a sibling node that lies within the transmission range of S_i has already been examined in the preliminary parent selection phase and does not need to be considered.

[5] Alternative choices are equally plausible. For example, prefer the nodes with the highest/lowest identifiers depending on whether this is an odd/even invocation of the aggregation tree formation algorithm.

- $AC_{i,p} \leq AC_{p,i}$. 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 aggregation tree may periodically get updated because of a significant change in the data distribution. 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.

5.2 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 $P_i + P_j - P_iP_j$, 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 $S_{j-1}, S_{j-2}, \ldots, S_1$.

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:

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

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.

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 [16]. Thus, the size of the transmitted data is independent of the number of nodes in the subtree, and 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 4.2, where the E_{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:

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

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.

A final and important note that we need to make at this point involves the estimation of the attachment cost when seeking to minimize the overall energy consumption in all the types of queries discussed in this paper. When each sensor node S_i examines the invitations of neighboring nodes (and only in this step) and estimates the attachment cost to any node S_j, in our implementation it utilizes the same E_{RF} value in order to determine the value of $E_{tr_{i,j}}$, independently on the distance of S_i to S_j. This is done so that the value of $E_{tr_{i,j}}$ is the same for all candidate parents of S_i, as desired by the proof of Theorem 1 in order to guarantee the lack of loops in the formed aggregation tree.

Theorem 1 *For sensor networks that satisfy the connectivity requirements of Section 4.1 our algorithm always creates a connected routing path that avoids loops.*

Proof:

We only sketch the proof here. It is obvious that any node that will receive the query announcement will select some node as its parent node. We first demonstrate that

no loops can be introduced and prove this by contradiction. Assume that the parent relationships in the created loop are as follows: $S_1 \rightarrow S_2 \rightarrow \ldots S_p \rightarrow S_1$. Let D_i be the distance (in number of hops) of node S_i from the `Root`. Since each node can select as its parent node a node with equal or lower D value, the existence of a path from S_1 to S_2 means that $D_2 \leq D_1$ and the existence of a path from S_2 to S_1 means that $D_1 \leq D_2$. Therefore, $D_1 = D_2$.

The attachment cost $AC_{i,j}$ calculated using the aforementioned statistics is of the form: $P_i \times (a_i + CF_j)$, where a_i is a constant for each node S_i. Considering that S_1 selected S_2 as its parent and not S_p, we get: $AC_{1,2} \leq AC_{1,p} \Longrightarrow CF_2 \leq CF_p$. By creating such inequalities between the current parent and child of each node, summing these up (please note that because one of the nodes in the loop will exhibit the highest identifier, for at least one of the above inequalities the equality is not possible), we get that: $CF_1 + \ldots + CF_p < CF_1 + \ldots + CF_p$. We therefore reached a contradiction, which means that our algorithm cannot create any loops.

∎

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 value for the attachment cost if we ignore the node's epoch participation frequency. This decision is expected to minimize the attachment cost, if the sensor at some point starts observing events.

Minimizing Other Metrics. Our techniques can be easily adapted to incorporate additional minimization metrics. For example, the formulas for minimizing the number of transmitted bits can be derived using the formulas for the energy minimization for the corresponding type of query. 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).

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 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 [16]. 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

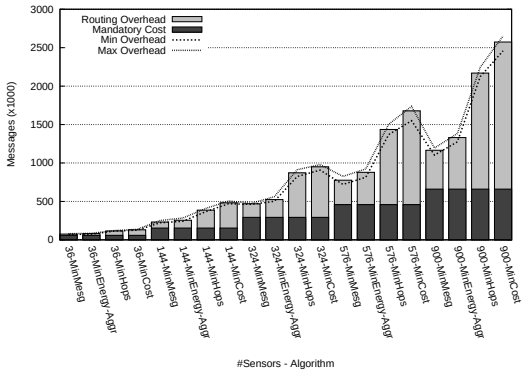

Fig. 1. Messages and Average Message Overhead for Synthetic Data Set.

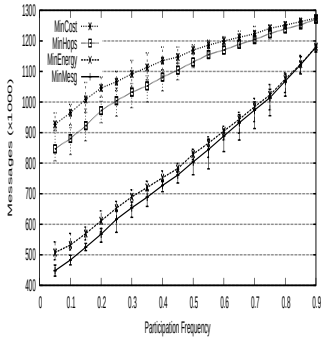

Fig. 2. Transmissions Varying the Epoch Participation Frequency

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 2. The message header was set to 32 bits, similarly to the size of each statistic and half the size of each aggregate value. In all figures we account for the overhead of transmitting statistics and invitation messages during the creation of the aggregation tree in our algorithms. All numbers presented are averages from a set of five independent experiments with different random seeds.

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. 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 SUM aggregate query over the values of epoch participating sensor nodes using all algorithms. We present the total number of transmissions for each algorithm and number of sensors in Figure 1. The corresponding average energy consumption by the sensor nodes for each case is presented in Table 4.

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 increase in messages induced by the *MinHops* and *MinCost* algorithms compared to our approach is up to 86% and 120%, respectively, with an average increase of 66% and 93%, respectively. However, since these gains depend on the number of transmissions that epoch-participating nodes perform, it is perhaps more interesting to measure the

Sensors	Aggregate SUM Query			
	MinMesg	MinEnergy	MinHops	MinCost
36	94.38	87.20	166.12	125.78
144	72.84	67.56	140.81	117.18
324	66.06	61.83	141.46	103.22
576	62.52	58.73	133.71	101.84
900	61.29	56.68	127.79	99.93
$\pm$	7.51%	10.94%	5.66%	6.8%

Table 4. Average Power Consumption (in mJ) for Synthetic Dataset with Error Bounds

# Sensors	MinMesg	MinEnergy	MinHops	MinCost
150	75.65	67.64	94.33	75.51
600	61.19	51.10	84.67	58.74
1350	58.87	47.86	85.89	55.48
$\pm$	9.58%	5.5%	15.01%	17.77%

Table 5. Average Power Consumption (in mJ) for SchoolBuses Dataset with Error Bounds

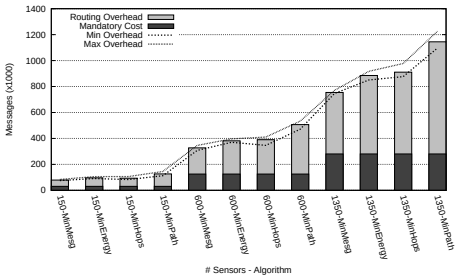

Fig. 3. Transmissions - Trucks data

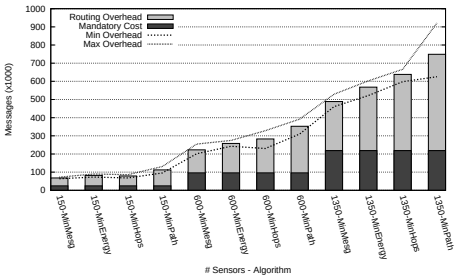

Fig. 4. Transmissions - SchoolBuses data

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 1, 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. The MinEnergy algorithm performs very well in this experiment. Compared to the MinHops algorithm, it achieves up to a 2-fold reduction in the power drain. Compared to the MinCost algorithm the energy savings are smaller but still significant (i.e., up to 76%).

We expect that the more the epoch participation frequencies of sensor nodes increase, the less likely that out techniques will be able to provide substantial savings compared to the MinHops and MinCost algorithms. In Figure 2 we repeat the aggregate query of Figure 1 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 2 validates our intuition, it also demonstrates that significant savings can be achieved even when sensor nodes have large P_i values (i.e., $P_i \geq 0.5$).

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 3 and 4 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. We present the average energy consumption of the sensor nodes in the same experiment for the SchoolBuses data set in Table 5. 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 15% more energy than our MinEnergy algorithm. Moreover, both our MinMesg and MinEnergy algorithms significantly reduce the amount of transmitted messages by up to 31% and 53% when compared to the MinHops and MinCost algorithms, respectively.

7 Conclusions

In this paper we presented algorithms for building and maintaining efficient aggregation trees in support of event monitoring queries in wireless sensor networks. We demonstrated that is it possible to create efficient aggregation 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. Furthermore, our techniques utilize a novel 2-step refinement process that significantly increases the quality of the created trees. In our future work, we plan to extend our framework to also support holistic aggregates and SELECT * queries, as well as extend our framework for a multi-query setting.

References

1. Rtree Portal. 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. J.-H. Chang and L. 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. Demers, J. Gehrke, R. Rajaraman, N. Trigoni, and Y. Yao. The Cougar Project: A Work In Progress Report. *SIGMOD Record*, 32(4):53–59, 2003.
9. A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-Driven Data Acquisition in Sensor Networks. In *VLDB*, 2004.
10. M. Duckham, S. Nittel, and M. Worboys. Monitoring Dynamic Spatial Fields Using Responsive Geosensor Networks. In *GIS*, 2005.
11. D. Estrin, R. Govindan, J. Heidermann, and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *MobiCOM*, 1999.
12. J. Gao, L. J. Guibas, N. Milosavljevic, and J. Hershberger. Sparse Data Aggregation in Sensor Networks. In *IPSN*, 2007.
13. C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidermann. Impact of Network Density on Data Aggregation in Wireless Sensor Networks. In *ICDCS*, 2002.
14. Y. Kotidis. Snapshot Queries: Towards Data-Centric Sensor Networks. In *ICDE*, 2005.
15. Y. Kotidis. Processing Proximity Queries in Sensor Networks. In *Proceedings of the 3rd International VLDB Workshop on Data Management for Sensor Networks (DMSN)*, 2006.
16. 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.
17. 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.
18. C. Olston and J. Widom. Offering a Precision-Performance Tradeoff for Aggregation Queries over Replicated Data. In *VLDB*, 2000.
19. S. Pattem, B. Krishnamachari, and R. Govindan. The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks. In *IPSN*, 2004.
20. V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2), 2002.
21. A. Sharaf, J. Beaver, A. Labrinidis, and P. Chrysanthis. Balancing Energy Efficiency and Quality of Aggregate Data in Sensor Networks. *VLDB Journal*, 2004.
22. A. Silberstein, R. Braynard, and J. Yang. Constraint Chaining: On EnergyEfficient Continuous Monitoring in Sensor Networks. In *SIGMOD*, 2006.
23. 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.
24. N. Trigoni, Y. Yao, A.J. Demers, J. Gehrke, and R. Rajaraman. Multi-query Optimization for Sensor Networks. In *DCOSS*, 2005.
25. W. Xue, Q. Luo, L. Chen, and Y. Liu. Contour Map Matching for Event Detection in Sensor Networks. In *SIGMOD*, 2006.
26. Y. Yao and J. Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. *SIGMOD Record*, 31(3):9–18, 2002.
27. W. Ye and J. Heidermann. Medium Access Control in Wireless Sensor Networks. Technical report, USC/ISI, 2003.
28. D. Zeinalipour-Yazti, P. Andreou, P. K. Chrysanthis, G. Samaras, and A. Pitsillides. The Micropulse Framework for Adaptive Waking Windows in Sensor Networks. In *MDM*, pages 351–355, 2007.