University of Alberta

Processing Spatiotemporal Data Map Queries with Redundancy Removal in Sensor Networks

by

Alexandru Coman Mario A. Nascimento JörgSander

Technical Report TR 05-23
September 2005

DEPARTMENT OF COMPUTING SCIENCE
University of Alberta
Edmonton, Alberta, Canada

Processing Spatiotemporal Data Map Queries with Redundancy Removal in Sensor Networks*

TR05-23

Alexandru Coman Mario A. Nascimento Jörg Sander
Department of Computing Science
University of Alberta
Edmonton, Canada
{acoman,mn,joerg}@cs.ualberta.ca

Abstract

Wireless sensor networks are made of autonomous devices that are able to collect information, store it, process it and share it with other devices. Such framework can be used to efficiently query spatiotemporal data, e.g., for monitoring humidity and temperature levels across a wide geographical region. Typical spatiotemporal region queries require the answers of only the subset of the network nodes that fall into the spatial area of the query. If the network is redundant in the sense that nodes' measurements can be substituted by those of other nodes with a certain degree of confidence, then only a much smaller subset of nodes may be sufficient to answer the query at a much lower energy cost. In this paper we investigate how to take advantage of such data redundancy, and we propose three techniques to process spatiotemporal region queries under these conditions. We show, through extensive experimentation, that taking advantage of the data redundancy reduces up to twenty times the energy-cost of query processing, thus prolonging the sensor networks lifetime.

1 Introduction

Wireless sensor networks consist of nodes with the ability to measure, store, and process data, as well as to communicate wirelessly with nodes located in their wireless range. There are many application domains where sensor network are well suited [22], e.g., environmental monitoring, warehouse management, traffic organization and surveillance. Sensors are typically battery operated, and since query processing unavoidably involves energy-wise costly communication among nodes, it imposes hard constraints on their lifetime. Therefore, energy-efficient query processing techniques are of utmost important within a sensor network, and that is the very focus of this paper.

In domains such as GIS [2], a typical query involves building a map of values for a given area, e.g., "find the temperature for each point of lot X12 at 2pm yesterday." Since it is not practical to have a sensor in each point of the monitored region, one has to settle for some approximated value for those points where a sensor is not present. This leads to the notion of an approximate answer for the same query where a map is built with values for each point in the map, but with also a confidence level for each measured value. In practice, this leads to two maps, one with the request values and another one

* This work is supported in part by NSERC Canada.

with the confidence associated with those values. In a more general case, the values of interest could form a set of maps, for different time granularities, e.g., "find the approximate hourly temperature values for the past 12 hours of lot X12 so long as the reported value has a confidence above 40%". We call this type of query a SpatioTemporal Data Map (STDMap) query. An important fact to note is that in the same way some points of the queried map are "covered" by one or more sensors, sensor themselves can also be covered by other sensors. In effect this means that some nodes may take advantage of such coverage redundancy and not participate in the query's processing without loss of the answer's quality. In this paper we exploit this redundancy in order to further extend the lifetime of the network, while still providing an effective means to query the network.

We focus on energy-efficient processing of STDMap queries over historical sensor data. As sensor nodes spend most of their energy during communication [1, 18], we aim at minimizing the amount of data exchanged among nodes during query processing. We study this problem in a peer-to-peer sensor network where all sensor nodes have similar capabilities, sensed measurements are stored locally, each sensor is only aware of the existence of the other sensors located within its wireless communication range, and the query can be initiated at any sensor. The advantages of this environment are network robustness, a balanced use of sensors' energy resources (since there is no centralization point), and a wide range of application scenarios where the presented solutions can be used.

An application domain where such a query and sensor network environment fits well is the environmental monitoring. The sensor nodes could be deployed from a plane over a region of interest. Upon activation, each node starts observing periodically various phenomena, such as the temperature and humidity of the soil. Park rangers patrolling through the region can access the network through any node in their proximity using a laptop. For instance, when certain events such as vegetation diseases or small fires are observed, the ranger could query the network about historical observations, which may help the ranger understand what has caused such events or learn about other areas that are threatened by similar events.

Our main contribution in this paper is the proposal of three techniques to address the problem of approximate query processing in sensor networks, namely: the AFM technique which employs parallel flooding of the queried region where each node decides, based on sound criteria, whether it should participate or not in the query's answer; the EFM technique, which is a energy-aware parallel flooding, where nodes decide about their participation considering also the amount of energy they have; and finally, MSM, which is a technique that uses the completion of the queried map itself as a guide to traverse the region's nodes. Note that all three techniques aim at taking advantage of the redundancy mentioned earlier, i.e., process the query so that not all relevant node need to participate (depending on the required level of confidence). Through extensive experimentation, we show that in-network processing of STDMap queries reduces by up to twenty times the energy use compared to the typical solution that retrieves the raw sensor measurements and assembles the STDMap answer off-line.

The remainder of the paper is organized as follows. Section 2 describes research related to ours. Section 3 introduces the STDMap query. Section 4 presents the characteristics of the peer-to-peer sensor network environment and introduces three algorithms for processing STDMap queries in this environment. Section 5 presents the experimental evaluation and Section 6 concludes the paper.

2 Related Work

Query processing with approximate answering in sensor networks has raised much interest from the research community due to its potential to substantially reduce the query processing costs. In [9]

the query answers are estimated using a statistical model for the sensors' readings, where the model captures the redundancy and correlation in sensor measurements. The sensors are interrogated just to help refine the model when the uncertainty is high, which reduces substantially the query processing costs. Sharaf et al. [23] exploit the temporal redundancy in a sequence of sensor readings to reduce the energy cost of aggregation during query processing. To improve the fault tolerance of query processing for aggregations, duplicate insensitive sketches are used in [5] to produce accurate approximations of the aggregate answers. In [6], the authors exploit the correlation and temporal redundancy among the readings of each sensor to compress the short-term historical measurements. Once compressed, the measurements are transmitted to a base station for long-term storage. Caching the sensors' readings is used in [10] to reduce the cost of retrieving the sensor data, with the users specifying their tolerance for stale data in the query. This paper complements the previous works in three directions: (1) we investigate the processing of historical queries over the sensor data stored at the nodes, (2) we study this problem in a peer-to-peer sensor network, (3) we exploit the spatial redundancy of sensor readings created by dense sensor network deployments, which allows us to use only a subset of nodes to answer the query.

Directed Diffusion [15] investigates query processing in a sensor network environment similar to ours in the sense that the query can be originated at any node, and nodes are only aware of their neighborhood. Differently from us, nodes do not store historical data and sensing is only performed in response to a query request. A system focusing on query processing over historical data is DIMENSIONS [12], which focuses on multi-resolution summarization of data for data mining. Several data-dissemination methods are discussed in [21], and the GHT system for data-centric storage is introduced. The simulation results show that the local storage of measurements performs the best for scenarios like ours where a large number of observations is available with only a small subset of them being retrieved. In [17, 18], the authors focus on acquisitional query processing in a sensor environment where information about all sensors is available at a base station. The base station creates and disseminates the query plan, with the sensors sensing the environment and transmitting the raw data upon request. The correlation in sensors' measurement is exploited in [8] to generate conditional query plans, where low cost attributes are used to determine the best plan for acquiring the high cost sensor measurements. The Cougar project [7, 26] also investigates techniques for query processing over sensor data. A central optimizer has the tasks of building a query plan and disseminating it to the relevant sensor nodes.

3 The STDMap Query

A common representation of information in environmental remote sensing [2] is in the form of a map capturing the spatial distribution of data (Figure 1), where each map point represents a spatial area and its associated value represents the state of the observed phenomenon in the area corresponding to the point. Query support for such a representation is important for applications where the spatial distribution of data is more important than individual data values. The map representation for sensor network data can be constructed by first collecting the sensor measurements from all sensors located in the region of interest, followed by the construction of the map off-line. However, collecting the measurements from all these sensors (called relevant nodes) may not be necessary, as the measurements from only a *subset* of the relevant nodes may be sufficient to construct the map. This is possible if the answers of some nodes can be approximated by the answers of other nodes. There are two reasons for considering answer approximation. First, it is not practical to have a sensor in each point of the monitored region, and therefore the values used for most of the map

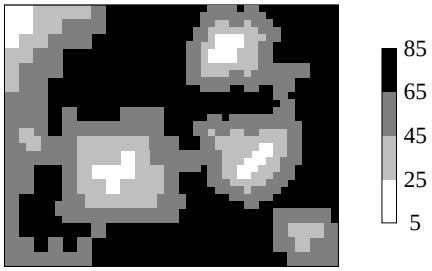

Figure 1: Example of a map answer

points must be approximated[1] using the answers of the sensors located nearby. Second, due to the inherent correlation among the states of physical phenomena at close locations, the measurement of any sensor can be approximated by the measurement of other sensors located nearby with a certain degree of confidence.

Let us consider two locations where we are interested in the state of a monitored phenomenon: location s where a sensor node S exists and location l where there is no sensor. S can provide a measurement for the monitored phenomenon only at location s. For location l, we can approximate a state with the measurement for location s, with some degree of confidence. We model the confidence of a node as a function $C(s, l)$, which represents the confidence of the sensor S that its measurement taken at location s is the same at location l. Depending on the monitored phenomenon and the capabilities of the sensing unit, the function $C(s, l)$ could be very simple or highly complex, constant over the lifetime of the sensor node or adaptive to various conditions (e.g., time of day). Naturally, the confidence of S for a location decreases with the distance to that location. The concept of confidence is also used in [9] to capture uncertainty, but, differently from us, their confidence represents the uncertainty with respect to approximated sensor readings, while ours captures the uncertainty in the validity of an actual sensor reading for a different spatial location than where it was acquired.

The SpatioTemporal Data Map query supports the map representation of the sensor network data. Assuming a confidence function *C(s,l)* which is dependent of the sensor network setting but it query-independent, we denote the query by STDMap*(qID,sw,tw,ct)*, with the following characteristics:

- It is identified in the sensor network using a unique query identifier *qID*.
- The answer of a STDMap query is a set of map layers representing the approximations of the sensor measurements with confidence above the minimum confidence *ct* for the region *sw*, with each layer corresponding to one time point within the query's time window *tw*.
- Each point of a map layer corresponds to a spatial area within the query's spatial window, with its value equal to the approximated state of the monitored phenomenon in the corresponding area. Figure 1 shows an example of a map layer.

Note that each sensor node has a confidence in approximating the state of the monitored phenomenon with its measurement for every possible location. Unfortunately, interpolating the measurements of all sensors for each location using their confidences is very difficult, possibly requiring information from every sensor node. Distributed regression is used in [13] to model spatiotemporal redundancy in sensors' measurements, where the user is responsible for providing the location of

[1]In this work we consider sensors that observe the state of a monitored phenomenon at the sensor location only. This is different from range sensing (e.g., movement sensing used in tracking [11]), which measures the state of an entity not necessarily located at a sensor's position, but in its vicinity.

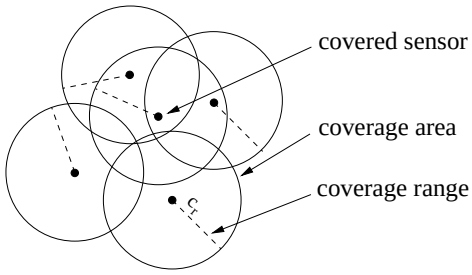

Figure 2: The coverage of sensors

kernels and the set of basis functions. This is not feasible for large sensor network deployments. In this paper we associate with a map point the measurement with the highest confidence among all approximations obtained during query processing, reserving the problem of interpolating the sensors' approximations for future work.

Before going further let us introduce the following definitions which are necessary for the remainder of the paper:

Definition 1. *Given a confidence function and a confidence threshold, the coverage area of a sensor node is the area around the sensor in which the sensor's data is relevant to the query (i.e., the confidence of the sensor is above the threshold for every point in the area).*

Definition 2. *Assuming the confidence function is uniform for all directions*[2] *from a sensor, the coverage range c_r of a sensor node is the radius of the circle centered at the sensor which forms its coverage area.*

As sensors' coverage areas can overlap (dependent on the inter-sensor distance, the confidence function and the confidence threshold), the coverage areas of some sensors may be covered by other sensors (see Figure 2 for an example). In this situation, the STDMap query can be answered using a subset of the relevant sensor nodes, thus saving communication and processing costs. This is possible as the STDMap query does not require in its answer the measurement with the highest possible confidence, but with a confidence higher than the confidence threshold *ct*. Note that a sensor's coverage area depends on both the confidence function $C(s, l)$ and the query (via *ct*). This is different from the sensing area that some range sensor types (i.e. motion sensors) provide, where the sensed area is a characteristic of the sensing device.

4 Strategies for Query Processing

We consider a peer-to-peer sensor network with fixed nodes that have equal roles in the functionality of the network. A query can be introduced into the network through any of the sensor nodes, with query answers located in some (possibly all) of the nodes. Due to the wireless network characteristics, a sensor node can communicate directly only with the sensors located within its wireless range, which form its neighborhood. A node can address a message to one of its neighbors (unicast) or it can address the message simultaneously to all its neighbors (broadcast), and it can communicate with nodes other than its neighbors using a multi-hop routing protocol. We assume that each node knows

[2]While the variation of physical phenomena may be different on each direction due to the environment, sensor nodes are not able to detect this and adjust the confidence function accordingly.

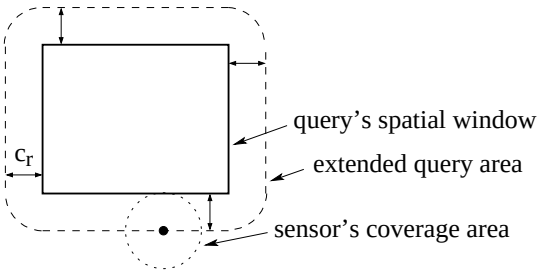

Figure 3: Extended query area

its location (e.g., it may use GPS during node activation), as well as the location of its neighbors (collected during network activation). Sensors take measurements periodically, and the collected values are stored locally for future querying. Each measurement has attached to it a time-stamp corresponding to the time of measurement. A sensor node with one megabyte of memory measuring humidity once every five minutes could store more than one year of raw data, which is beyond the expected lifetime of some of the sensor devices currently available. If high sensing rates are required or long-term storage is expected, one could adopt data stream storage solutions for fixed storage space such as those proposed in [27].

A major constraint on sensor nodes is their limited energy supply. Since the energy required by sensing and computation is up to three orders of magnitude less than the energy used for communication [19, 18], we are interested in the energy cost of communication during query processing.

4.1 Processing the STDMap Query

Since the coverage areas of the sensors located in the proximity of the query's spatial window may intersect the query window, finding the query answer for a location inside the query window may require contacting nodes located outside the query window.

Definition 3. *The extended query area is the area where the sensor nodes whose coverage areas intersect the query's spatial window can be located.*

Algorithms for processing STDMap queries must be able to contact the nodes located in the extended query area. Given a confidence function and a confidence threshold, the extended query area is formed by the extension of the query's spatial window in every direction with the coverage range, as shown in Figure 3.

Due to the nature of the environment where the sensors are deployed, it is possible that the approximations of two neighboring sensors for the same location are inconsistent. This suggests possible obstacles affecting the expected variation of the monitored phenomenon. In this situations nodes could mark the affected areas as not covered by their neighbor that provides the inconsistent approximation. We leave the problem of inconsistent approximations for future work, and we assume in this paper that the nodes covering a certain location provide consistent approximations for the location.

4.2 Algorithms for Processing STDMap Queries

This section presents three algorithms for processing STDMap*(qID,sw,tw,ct)* queries. In large scale sensor deployments, the user is typically interested at a given time in a spatial window (*sw*) of the

whole monitored region and a temporal window (tw) of the measurements collected by a sensor. In [4], it has been shown that for spatiotemporal region queries a two-phase query processing approach is more efficient that the typical network flooding. By forwarding the query to all network nodes, the network flooding contacts many nodes in addition to those that hold the query answers, which increases substantially the energy cost of processing.

Due to its lower cost, we use a similar two-phase approach in this paper. We break each processing algorithm into two phases: one for finding a routing path from the query originator node to the query window sw, the other for collecting the query answers from the relevant nodes and returning the answers to the originator node. While all three proposed algorithms differ in their processing strategy for the second phase, they use the same routing algorithm for the first. We use a simple greedy approach to discover a routing path from the query originator node to a node located near the center of the query's spatial window, called coordinator node. At each step of the route discovery, the current node forwards the query to its neighbor located closest to the center of the query window. Greedy-based routing methods for position based routing have been shown to nearly guarantee delivery for dense network graphs [25], as it is the case for sensor networks [24]. If the sensor network is not dense, more advanced geographic routing techniques such as GPSR [16] (slightly modified as in [21]) could be used to improve the route discovery for the first phase. Note that if the query originator node is located inside the query's spatial window, which includes the case where the query window covers the whole network, the first phase is not required.

As the proposed processing algorithms use the the same routing solutions in their first phase (if required), we present in the following only the second phase for each query processing algorithm.

4.2.1 Aggressive Flood Strategy (AFM)

For its second phase, the Aggressive Flood for STDMap (AFM) algorithm uses parallel flooding to distribute the query to all nodes located within the extended query area. However, not all nodes will contribute to the answer. Each node makes a decision locally if its answer is required by the coordinator node to assemble the STDMap answer. The decision is based on the coverage areas and the states of its neighbor nodes. Each one of a node's neighbors can be in any of the three states: OPEN, if the node has no information about its neighbor's state; SEND, if the neighbor has decided that its answer is required; or SKIP, if the neighbor has decided that its answer is not required. A node decides its state (SEND or SKIP) after receiving broadcasts of the query from one or more of its neighbors, but before sending its own query broadcast. Nodes send their state (SEND or SKIP) along with the query broadcast. Once a node makes a decision, it can filter out the other broadcasts of the same query based on their header. A node decides to send its answer and changes its state to SEND if its coverage area is not fully covered by its neighbors in OPEN or SEND state. Otherwise, the node does not send its answer and changes its state to SKIP, as other nodes will cover its area. A node that decides its answer is required (SEND state) returns its answer to the neighbor it first received the query from. The second phase of the AFM algorithm running in each relevant node is presented in Algorithm 1.

Using AFM, each node receives the query message from several neighbors and sends one broadcast. Depending on the overlap of the coverage areas, only a subset of the relevant nodes will return their answers to the query coordinator node. After the query coordinator gathers the nodes' answers, it constructs the STDMap answer and returns it to the query originator node over the path discovered in the first phase of query processing. Due to the limited information each node has, if a node is covered with the help of other nodes than its neighborhood, the overlap is not detected. As the query is forwarded in parallel over all paths, each node is reached over the shortest path from

Algorithm 1: AFM Algorithm - Phase 2

```
   Input  : Node N, NeighborList NB
 1 Receive query Q, PNB.state from ParentNeighbor PNB
 2 if N.location not in Q.extArea then STOP
 3 Initialize status of all nodes in NB to OPEN state
 4 Update status of node PNB in NB to PNB.state
 5 Construct coverage N.cov using N.location, Q.extArea, Q.ct
 6 Check if nodes in NB with SEND or OPEN states cover N.cov
 7 while N.cov is covered & new broadcasts received do
 8 |   Update status in NB for broadcasting neighbors
 9 |   Check if nodes in NB with SEND or OPEN states cover N.cov
10 if N.cov is covered then
11 |   N.state ← SKIP /* N's answers are not required */
12 |   Broadcast Q, N.state
   else
13 |   N.state ← SEND /* N's answers are required */
14 |   Broadcast Q, N.state
15 |   Return (N.data in Q.tw) to PNB /* return answer */
```

the coordinator. Thus, the answers are returned to the coordinator node over the shortest path by the nodes that decide their answer is required.

When a node changes its state to SKIP, it may force its neighbors in OPEN state to cover its area. Assuming no communication delay latency, these neighbors must have smaller or equal hop-counts from the coordinator node, and therefore the algorithm allows nodes closer to the coordinator node to skip answering and force their neighbors farther away to cover their area. For the AFM algorithm to function correctly, it must be implemented on top of a wireless protocol using control frames and virtual channel sensing (such as [3]). Such protocols ensure that neighboring nodes do not broadcast their messages at the same time[3]. If such protocol is not used, two neighbors that count on each others coverage could broadcast the query and their decision simultaneously. It would be too late for the two nodes to change and re-broadcast their decisions, as their non-common neighbors reached by the first broadcast may have already processed the original decisions. The AFM algorithm is an aggressive strategy, in the sense that a node decides to skip answering with only partial knowledge of its neighbors decision, thus forcing other nodes to answer on its behalf. The correctness of the AFM algorithm is stated in the following lemma:

Lemma 1. *Any area from the query's spatial window covered by sensors will be covered in the final answer using the* AFM *algorithm.*

Proof. We assume the network within the extended query area is connected and every area within the query's spatial window it covered by the coverage areas of one or more sensor nodes. Let us assume there is an area A from the query's spatial window that is covered by nodes $N_1, ..., N_k$, and only those, and A is not covered in the answer, i.e., none of the nodes $N_1, ..., N_k$ is sending its answer to the coordinator node. Thus all nodes $N_1, ..., N_k$ are in SKIP state. However, there is N_j ($j \in 1...k$) such that N_j broadcasts the query and its status the last among its neighbors covering A. Node N_j

[3]Before a node is able to reserve the communication channel to broadcast the query and its state, the channel may be used by some of its neighbors for their broadcasts, which the node will receive and consider in its decision.

must have received broadcasts from all its neighbors covering A before deciding its status. Thus node N_j knows that its neighbors covering A are in SKIP state (all neighbors covering A must be among $N_1, ..., N_k$), so node N_j cannot find any neighbor to count on for the coverage of A. Therefore, node N_j must be in SEND state, and area A is covered in the final answer. □

4.2.2 Energy-aware Flood Strategy (EFM)

The Energy-aware Flood for STDMap (EFM) algorithm uses also a flooding strategy in its second phase, but, differently from the AFM algorithm, its correct execution does not require any specific support from the wireless network protocol used in the sensor network. While AFM is an aggressive strategy, EFM is a passive strategy, where a node makes a decision aware of the state of its neighbors. For the state information of all neighbors to be available, a node must receive all the broadcasts of its neighbors and send two broadcast messages to provide additional information about its status. While all three proposed algorithms try to reduce the energy used during query processing, EFM uses information about the amount of energy nodes have in order to decide which nodes should participate in query answering. Thus, EFM is an energy-aware processing strategy.

Query processing starts when the coordinator node broadcasts the query to its neighbors regardless of their state. A node receiving a query for the first time checks if its coverage area is covered by its neighbors. If it is not covered, the node decides that its answer is required and sets its state to SEND. If the area is covered, the node does not have yet sufficient information about its neighbors to safely decide to skip answering, and therefore sets its own state to OPEN. Next, the node broadcasts the query and its current state.

Once a node has received the query broadcast from all its neighbors, it checks if its neighbors that have decided to answer are covering its area, i.e., if its area is covered by nodes in SEND state. If its area is covered, the node can safely skip answering and it sets its state to SKIP. After this check, each node broadcasts a state update message to its neighbors and waits for the state updates from its neighbors. If a neighbor B of a node A has changed its state from OPEN to SKIP, it means that B is fully covered by its neighbors in SEND state, and thus any overlap it has with A is also covered. Consequently, a node can safely skip answering if its area is fully covered by its neighbors in SEND or SKIP states. Such a node changes its state to SKIP, but this information is not exchanged. Not exchanging the information on this status update is safe, as any node that is counting on the coverage of its neighbor in SKIP state remains covered (through transitivity) by the nodes covering its neighbor.

At this point, a node can be in any of the three possible states (SEND, SKIP or OPEN). If a node's state is SEND, the node returns its answer to the neighbor it first received the query from. If its state is SKIP, nothing has to be done as the node's coverage area is covered by other nodes that will answer. If its state is OPEN, the node must decide based on the information about its neighbors whether to send or to skip. To determine correctly whether it has to send its answer, however, it needs to know if information about its own area that is covered by neighbors that are also in OPEN state will be sent (by the neighbors directly or by other nodes covering the neighbors). This is a potential problem as the covering relation is symmetric[4]. Two nodes in OPEN state which partially cover each others areas have to independently make a consistent decision. In particular we must avoid that two nodes decide to skip and no other node will send information about their overlapping area. The information about the current state of its neighbors is not sufficient to take a consistent

[4]If the coverage areas of two neighbors overlap, each node may consider the other node covering the overlapping area.

decision.

We use the amount of energy left in nodes to determine which one of a pair of nodes in OPEN state with overlapping coverage areas will ensure the coverage of the overlap (the energy information can be exchanged during the broadcasting of the state update message). For each pair, the node with more energy will be responsible for the coverage of the overlapping area. Thus, a node A in OPEN state with more energy will not count on a neighbor B in OPEN state with less energy to cover its area, and vice versa, B will count on the coverage by A, i.e., B will consider A to be in SEND state. This policy guarantees that for each pair of neighboring nodes in OPEN state with overlapping areas only one node will assume that the overlap is covered by the other node. If two nodes have the same amount of energy left, they can use their unique node identifier as a tie-breaker.

Algorithm 2: EFM Algorithm - Phase 2

```
   Input  : Node N, NeighborList NB
 1 Receive query Q, PNB.state from ParentNeighbor PNB
 2 if N.location not in Q.extArea then STOP
 3 NBE ← ∅ /* list of neighbors in Q's extended area */
 4 foreach node N_i in NB do
 5 |   if N_i in Q.extArea then
 6 |   |   Add N_i to NBE
 7 Initialize status of all NBE nodes to OPEN state
 8 Update status of node PNB in NBE to PNB.state
 9 Construct coverage N.cov using N.location, Q.extArea, Q.ct
10 Check if all NBE nodes cover N.cov
11 if N.cov is covered then
12 |   N.state ← OPEN
   else
13 |   N.state ← SEND
14 Broadcast Q, N.state /* SEND|OPEN state */
15 Wait for broadcasts of Q, status from all NBE nodes
16 Update status for all NBE nodes based on broadcasts
17 Check if NBE nodes with SEND state cover N.cov
18 if N.cov is covered then
19 |   N.state ← SKIP
20 Broadcast N.state, N.energy /* SEND|OPEN|SKIP state */
21 Wait for broadcasts of status, energy from all NBE nodes
22 Update status, energy for all NBE nodes
23 foreach node N_i in NBE do
24 |   if N_i.state = OPEN & N_i.energy > N.energy then
25 |   |   Change status of N_i in NBE to SEND state
26 Check if NBE nodes in SEND or SKIP state cover N.cov
27 if N.cov is not covered then
28 |   Return (N.data in Q.tw) to PNB
```

After each node updates its local representation of the state of the OPEN neighbors according to the above policy, it checks again if its area is covered by neighbors in SEND or SKIP states. If its area is not covered, the node will assume its answer is required and sends it to the coordinator node. Otherwise, if its area is covered, it skips answering. Note that if a node A skips answering,

this does not affect a neighbor B with less energy that counts on A for covering their overlap, since other nodes will answer for A's area. The pseudo-code for the second phase of the EFM algorithm is listed in Algorithm 2, and the correctness is stated in Lemma 2.

Lemma 2. *Any area from the query's spatial window covered by sensors will be covered in the final answer using the* EFM *algorithm.*

Proof. We assume the network within the extended query area is connected and every area within the query's spatial window it covered by the coverage areas of one or more sensor nodes. Let us assume there is an area A from the query's spatial window that is covered by exactly the set of nodes $\{N_1, ..., N_k\}$, and A is not covered in the answer, i.e., none of the nodes $N_1, ..., N_k$ is sending its answer to the coordinator node. Thus nodes $N_1, ..., N_k$ are in SKIP state. However, there is $j \in 1...k$ such that N_j has maximal energy among $N_1, ..., N_k$. The node N_j cannot skip. There is no neighbor N^* of N_j covering A having higher energy so that node N_j could count on N^* to cover A (if such a node would exist, it would be among $N_1, ..., N_k$, and N_j would not be the node with the highest energy in that set). Therefore, node N_j must be in SEND state, and area A is covered in the final answer. □

4.2.3 Map-guided Search Strategy (MSM)

The third solution we propose for processing the STDMap queries is the Map-guided Search (MSM) algorithm. Differently from the previous two algorithms, MSM uses a partial query answer for guiding the query processing. The algorithm finds the STDMap answer by forwarding the query using the current partial STDMap answer. At each step, the current sensor selects for query forwarding its neighbor that can provide answers with confidence above the confidence threshold *ct* to the largest number of map points that do not hold an answer yet. If a neighbor of the current sensor has a coverage area where every location has already been covered with confidence higher than *ct*, then the neighbor is not contacted at all during query processing.

The query message received by a node contains both the query and the partial STDMap answer as obtained so far. The node first answers the query and adds its answers to the STDMap answer. Before forwarding the query to any neighbor, the neighbors that are not located within the extended query area are discarded from the list of candidate neighbors. The algorithm goes iteratively through the candidate neighbors and forwards the new partial STDMap answer to the best of them until either the query area is fully covered or there are no more candidate neighbors. The query and the STDMap answer is forwarded to the neighbor that covers most map points that do not have associated an approximated measurement value. The query answering is considered complete when STDMap's map layers have associated approximated values for every location. The correctness of MSM algorithm is stated in the following lemma:

Lemma 3. *Any area from the query's spatial window covered by sensors will be covered in the final answer using the* MSM *algorithm.*

Proof. By construction of the MSM algorithm. We assume the network within the extended query area is connected and every area within the query's spatial window it covered by the coverage areas of one or more sensor nodes. The MSM algorithm uses sequential depth-first forwarding to contact nodes and therefore all nodes within the query extended area are contacted until all of the query area is covered. □

Algorithm 3: MSM Algorithm - Phase 2

```
   Input  : Node N, NeighborList NB
 1 Receive query Q, map M from ParentNeighbor PNB
 2 Update M using N. location, N.data and Q
 3 CNB = ∅ /* candidate neighbors for forwarding */
 4 foreach node N_i in NB do
 5     if N_i.location in Q.extArea then
 6         Add N_i to CNB
 7 while CNB not empty & M not full do
 8     BN = ∅ /* best neighbor */
 9     foreach node N_i in CNB do
10         if gain(N_i, M) = 0 then
11             Remove N_i from CNB
12         if gain(N_i, M) > gain(BN,M) then
13             BN ← N_i
14     if BN ≠ ∅ then
15         Send Q, M to BN
16         Wait for M from BN
17         Remove BN from CNB
18 Return map M to PNB
```

Since the query forwarding is done in a sequential depth-first-like manner, only one node is processing the query at each point in time. This process ensures that only one copy of the partial STDMap answer is available in the network, and each node processing the query is aware of the contribution to the STDMap answer of the nodes previously involved in the query answering. While this strategy is likely to result in slower query answering for each individual query than the other two algorithms, it facilitates several queries being processed simultaneously by the same group of relevant sensors. The second phase of MSM algorithms is listed in Algorithm 3, where the *gain* function counts the number of map points covered by a node that don't have any approximated measurement associated yet.

4.3 Coping with Sensor Failure

Quite often sensor networks operate in harsh environments, where permanent or transient sensor failures are expected. In addition, the energy source of sensor nodes is limited, leading to their failure after a period of operation. Regardless of the reason, node failures should be rare events during most of the network lifetime, otherwise the use of sensor network technology is not practical. In the following we discuss solutions for dealing with node failures during the processing of STDMap queries.

We first consider the MSM algorithm, where the communication takes place only between two nodes at a time. When a node tries to send the query to one of its neighbors, the neighbor is expected to confirm receiving the query message. If the neighbor does not confirm receiving the query, the node assumes that its neighbor has failed and selects another neighbor for query forwarding. If the neighbor acknowledges receiving the query, the node periodically checks the availability of its neighbor until an answer is returned. If the neighbor appears unavailable during several successive

checks, the node assumes its neighbor has failed and selects another neighbor for query forwarding. This solution is consistent with the sensor network environment we assumed (in the sense that nodes are only aware of their neighbors) and it guarantees that the query answer will be found[5]. If a node fails during the first phase of query processing, the failure can be treated similarly, since the greedy solution we use in the first phase of all three algorithms also uses communication between two nodes at a time.

When flooding is used for forwarding the query (in the second phase of AFM and EFM algorithms), node failures cannot be addressed locally in the neighborhood of the failed nodes due the the parallel nature of the query processing. In this case, nodes detecting a failure should report the location of the failed node to the coordinator node. If the coordinator node does not receive any answers to the STDMap query for the regions where the failed nodes were located, it is possible that some of the neighbors of the failed nodes may have counted on the coverage of the failed nodes. Thus, the coordinator will restart the query processing in the query area. An optimization is for the coordinator node to restart the query processing only for the affected query area.

In both situations discussed above, it may happen that a neighbor has forwarded the query before failing and the query is already being processed in the network. Forwarding the query to a different neighbor or restarting the query processing at the coordinator node would cause the query to be processed twice, or even multiple times if several nodes fail after forwarding the query. This may not be desirable given the limited energy resources that sensors have. There are solutions for coping with nodes failure that trade the robustness of query processing for energy efficiency. We plan to investigate thoroughly solutions for sensor failure and their integration with the proposed algorithms in our future work.

5 Experimental Evaluation

We implemented a sensor network simulator to study the performance of the presented algorithms. The placement of the sensor nodes follows a uniform distribution over a two dimensional region. We represent a STDMap query by the coordinates of a spatial window (sw), a temporal window (tw), a confidence threshold (ct) and a query identifier (qID). The query's spatial window covers 4% of the monitored region (that is 20% on each spatial coordinate), unless otherwise noted. The query's temporal range covers 60 measurements (one hour of sensor measurements for a measurement rate of one per minute, or two months of measurements for one daily measurement). For the answer of the STDMap query, each point of a map layer corresponds to 1 m^2 square area in the query's spatial window. If a different map granularity is required, the granularity could be added as a parameter to the STDMap query. The query originator and the center of the query's spatial window are uniformly distributed over the monitored region. A summary of query and sensor network parameters and their default values used in our experimental evaluation is presented in Table 1.

While our algorithms are general with respect to the confidence function $C(s, l)$ used by the sensor nodes, an explicit confidence function is required in the evaluation. We use the Gaussian distribution function $C(s, l) = e^{\frac{-d(s,l)}{2*\sigma^2}}$ to model the confidence function, where $d(s, l)$ represents the Euclidean distance between the sensor node located at s and location l and σ^2 is the variance of the function. Gaussian functions have been used before [9] to capture the behavior of physical phenomena and the correlation in sensor measurements. Using this function, the confidence of a sensor is high in

[5]We assume that the network density remains high despite node failures. If the sensor network density decreases substantially, the network graph may become disconnected, making the network unusable.

Parameter	Default Value
Size of monitored region	1000x1000 m
Wireless range	50 m
Average number of neighbors	15
Size of a measurement tuple	64 bits
Spatial window (sw)	4% (of region)
Temporal window (tw)	60 measurements
Confidence threshold (ct)	0.40, 0.80
Size of query message	256 bits
Energy used to transmit a bit	$\alpha + \gamma d^n$ nJ/bit [20]
Energy used to receive a bit	β nJ/bit [20]

Table 1: Parameters of query and sensor network

its proximity and decreases rapidly with increasing distance. Given a confidence function $C(s, l)$ and a confidence threshold ct, a sensor's coverage area is determined by the confidence range c_r (see Section 3). When c_r is smaller than half of the wireless range, the sensors that cover a sensor's area are among its neighbors. When c_r is larger than half of the wireless range, the sensors that cover a sensor's area may be located outside its wireless range. We use $\sigma = 50$ and show results for two confidence thresholds $ct \in \{0.4, 0.8\}$, which allows us the evaluate the algorithms for both situations[6].

We compare the performance of the algorithms using two metrics. The first metric evaluates the algorithms for their capability to reduce the number of relevant sensors needed to answer the STDMap query. Finding a smaller number of nodes to answer the query may increase the energy cost of processing due to an increase in the communication cost for the control messages. Thus, we use the total energy used during query processing as our second metric. Since we are interested in energy-efficient query processing, we are mainly interested in the energy metric, but the first metric allows us to better understand the behavior and trade-offs of the algorithms. We use the following values for the parameters of the energy costs [14]: $\alpha = \beta = 50\ nJ/bit$, $n = 2$, and $\gamma = 10\ pJ/bit/m^2$. As typical sensors do not have sophisticated communication electronics capable of adapting the transmission range [7], all messages are transmitted as far as the wireless communication range. Similar to [21], our simulator considers that the message delivery is instantaneous and error-free between nodes communicating directly. The simulator faithfully captures the behavior of the algorithms and their relative performance. In our experiments we only measure the energy used to transmit and receive messages during query processing, which includes the messages used for query forwarding, for returning the answers and for status updates. We focus on the energy-efficiency of the query processing algorithms and make the measurements independent of the characteristics of the MAC layer (for instance 802.11 radios consume as much energy in idle mode as for receive mode, while other radios may switch to a low-energy state when idle).

To capture the relative performance of our algorithms against the typical query processing solution, we implemented a simple network flooding with spatiotemporal constrains, called STF. In STF, the query originator node sends the query to all the sensor nodes, but only those located in the extended query area answer the query, returning the raw sensor measurements collected within the query's time window to the originator. Since this solution does not use a coordinator node, all

[6]For $ct = 0.4$, $c_r \approx 44\ m$, and for $ct = 0.8$, $c_r \approx 22\ m$.

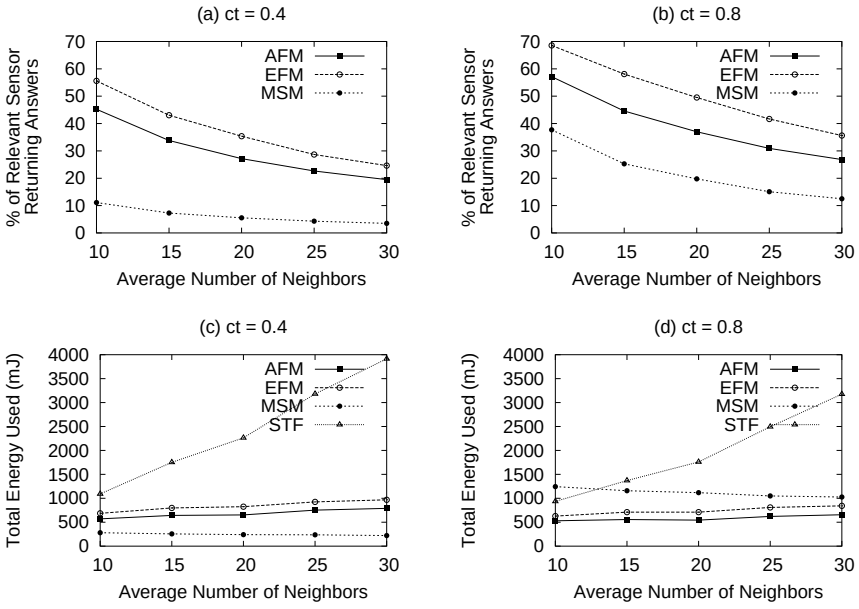

Figure 4: The effect of the number of neighbors

query answers are returned to the originator node over the shortest path. For consistency with the format of data collected by the STF algorithm, the proposed algorithms store the map layers as the raw measurements from which the layers can be constructed. This is reasonable as most values forming the map layers are approximations of a few measurements. In reality, the map layers could be stored as compressed images, which would substantially reduce their size.

5.1 Varying the Node Density

We use the average number of neighbors per sensor to represent the sensor network density. This parameter combines the size of the monitored region, the number of sensors and the wireless range. Studying the effect of the number of neighbors on each algorithm helps us understand the effect of each of these three parameters on the algorithms.

Figures 4(a) and 4(b) show the percentage of relevant nodes that answered the query for each algorithm. Note that for the STDMap query the relevant nodes are the nodes located within the extended query area. The STF method retrieves the raw measurements from all the relevant nodes (i.e., 100%), and therefore it is not shown in the graphs. The AFM and EFM methods forward the query to all relevant nodes, with only some of these nodes answering the query. The MSM algorithm contacts only the nodes that are used in answering the query. As the number of neighbors increases, all algorithms except STF select a smaller percentage of the relevant nodes to cover the query window. In the case of MSM, each node has a larger set of neighbors to choose from for the next step of the algorithm, which leads to a better selection of the nodes used for covering the query area. For the AFM and EFM algorithms, a larger set of neighbors increases the probability that a node's confidence area is covered, which reflects on the lower percentage of nodes that answer the query. The aggressive strategy of the AFM algorithm leads to a smaller percentage of nodes that answer the query compared to the energy-aware solution used in EFM. The increase in the confidence threshold reduces the coverage range, which has a double effect on the query processing

performance: both the extended query area and each sensor's confidence area are smaller. A smaller extended query area contains a smaller number of relevant nodes, but, on the other hand, the smaller coverage areas force more sensors to answer the query in order to cover the query area. Overall, the increase in confidence threshold forces a larger portion of the relevant sensors to answer the query in all our algorithms. This effect is amplified in the MSM algorithm because of the reduced overlap among the sensors that MSM uses for query processing. For the AFM and EFM algorithms, the overlap of sensors' coverage areas is high for $ct = 0.4$, which allows these methods to have a smaller increase than MSM in the number of relevant sensors used for answering the query when $ct = 0.8$.

Figures 4(c) and 4(d) show the variation of the total energy used for processing a query when the network density increases. The STF algorithm is the most affected due the linear increase in the number of relevant sensors that return answers with the increase in density. Our algorithms use only a few more relevant sensors to answer the STDMap query for denser networks, which reflects in their slow increase in the energy usage. The EFM algorithm uses two floods of the query window, which causes a slightly higher energy increase than for the AFM algorithm. On the other hand, the AFM algorithm forces nodes located farther away from the coordinator node to answer, which increases its energy cost for gathering the answers.

The MSM algorithm uses less nodes than the AFM and EFM algorithms to answer the query, which reflects on the its lower energy costs when $ct = 0.4$. When nodes' coverage areas are small ($ct = 0.8$) and the network density is low, MSM contacts a large percentage of the relevant nodes. This leads to a high energy cost for MSM, even higher than STF. There are two reasons: first, the cost of transferring the STDMap partial answer to every contacted node in order to keep track of the already covered area is not negligible; next, MSM uses a depth-first strategy, which leads to most nodes contacted to be on the same path, and thus the STDMap answer is returned to the coordinator node over a long and costly path. Overall, the AFM and EFM algorithms are the least affected by the increase in network density. While MSM uses the least energy for low confidence thresholds (up to twenty times less than STF) and the least number of sensors to answer the STDMap query, it is more sensitive to the network density, using more energy than STF for a combination of low density and high confidence threshold.

5.2 Varying the Size of the Query's Spatial Window

In the second set of experiments, we kept all other parameters constant, while varying the size of the query's spatial window between 2 and 10 percent of the size of the monitored region. Figures 5(a) and 5(b) show the effect of query size on the percentage of relevant sensors that answer the STDMap query. The AFM algorithm takes advantage of the increase in the query size and forces the nodes farther away to cover the nodes closer to the query, which leads to a slight decrease in the percentage of relevant nodes that answer the query. The EFM algorithm uses each node's neighborhood to decide which nodes should answer, and therefore it is not affected by the size of the query area and uses about the same percentage of relevant nodes to answer the query for all query sizes. In the case of MSM, larger query sizes are more difficult to cover efficiently, and thus the percentage of relevant nodes that answer the query increases with the query size. When the confidence threshold is higher, all algorithms except STF use a larger set of sensors to cover the query area since the area that each sensor covers is smaller. Consistent with our observation when investigating the effect of network density, higher confidence thresholds lead to a larger increase in the percentage of relevant nodes used by the MSM algorithms compared to the AFM and EFM algorithms.

Figures 5(c) and 5(d) show the total energy used during query processing for the investigated algorithm when the size of the query's spatial window is varied. The MSM algorithm is affected the

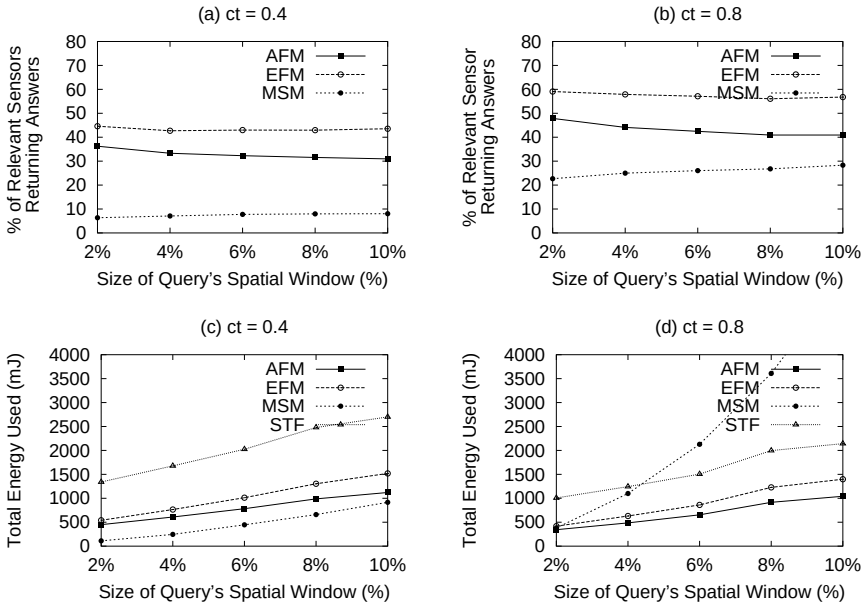

Figure 5: The effect of the query's spatial window

most by the increase in query size due to the the larger number of nodes that it contacts and the increased length of the path over which these nodes are contacted and the answers are returned. The increase in the query size also leads to an increase in the energy used by the flood-based algorithms as the number of relevant nodes increases and more nodes must answer the query to cover the larger query area. In addition, the EFM algorithm uses two floods over the relevant sensors for updating the sensors' status, which causes an increase in its energy usage compared to the AFM algorithm for larger query areas. Overall, the MSM uses the least energy when the number of relevant nodes answering the query is small, but it costs increases sharply with the increase in the number of these nodes (Figure 5(d)). The AFM and EFM algorithms behave better than MSM, the increase in the query area causing a smaller increase in their energy costs. In addition, they use two to five times less energy than STF for processing the STDMap queries.

5.3 Varying the Distribution of Queries

To test out intuition that the EFM algorithm produces a more balanced energy use at the nodes within the extended query area than the AFM algorithm, we compared the effect of the two methods on the nodes' energy levels over several executions of the same query. We fixed the position and size of the query's spatial window while we allowed the originator node to be randomly selected among the network's nodes. We only measured the energy used by the algorithms for collecting the answers in their second phase of the algorithms. Before starting the query processing, we charged all nodes with similar energy levels. After each set of 100 executions of the query, we calculated the average energy left in the nodes within the extended query area, and the standard deviation of energy from the average for the same nodes. We use the standard deviation to evaluate the energy balance among the sensor nodes.

Figure 6 shows the standard deviation of energy for the nodes within the extended query area. As more queries are processed, EFM produces a more balanced energy use compared to AFM. This

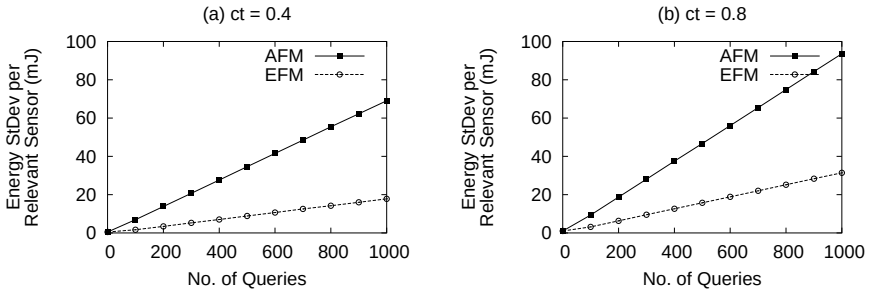

Figure 6: The effect of the query distribution

benefit of EFM is likely to extend the quality of the query processing in the long-run since more nodes will be available for a longer time. AFM forces some nodes to use more energy than their neighbors, which leads to their early failure, likely creating gaps in the network's coverage. With the increase in the confidence threshold, the difference in the energy balance of the two algorithms increases. While our previous experiments have shown that the AFM algorithm is slightly more energy-efficient than EFM, the EFM algorithm balances better the energy use across the sensor nodes, leading to an increased network lifetime.

6 Conclusions

While the technological advances have lead to sensors with reduced sizes and increased capabilities, the sensor data management is still in its infancy. The challenges are multiple, and the database research has to move its focus from considering time as a main optimization goal towards energy-efficiency or a combination of both time and energy. The size of the database is no longer a primary challenge, with the focus moving to the distributed nature of the database and query processing.

In this paper we proposed the STDMap query which exploits the redundancy of sensor measurements on the spatial dimension. We showed that the STDMap query can be answered using only a subset of the relevant nodes. We introduced three strategies for processing STDMap queries (MSM, AFM and EFM) in a peer-to-peer sensor network environment with fixed nodes, where nodes have equal capabilities, are only aware of their neighbors, and the query can be initiated at any node.

In an extensive experimental evaluation, we studied the performance of the proposed algorithms under several conditions. We showed that the AFM algorithm is the most energy-efficient for most scenarios, closely followed by EFM. While the AFM algorithm requires support from the protocol layer, the EFM algorithm is independent of the underlying protocol, which recommends it for most applications. A secondary benefit of the EFM algorithm is that it provides a balanced energy use at the sensor nodes, an important advantage in applications where the queries are not uniformly distributed. In all our experiments EFM has shown low energy usage and consistent performance with respect to various network and query parameters, and therefore we also recommend it for processing queries with similar characteristics to the STDMap query.

Our investigations have revealed several issues that require further analysis. The harsh environment where sensor networks typically operate raises the issue of sensors' transient or permanent failure. The trade-off between the robustness and energy-efficiency of query processing in the case of sensor failures requires further attention. The natural medium may also cause measurement inconsistencies among sensors and solutions for handling them are in our focus. In this paper we

considered as the most reliable approximation the one with the highest confidence. Combining the approximations of several sensors based on their confidence is an open problem that has potential for improving the quality of approximations.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):392–422, 2002.

[2] E.C. Barret and L.F. Curtis. *Introduction to Environmental Remote Sensing*. Stanley Thornes, 1999.

[3] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: a media access protocol for wireless LAN's. *SIGCOMM Computer Comm. Review*, pages 212–225, 1994.

[4] A. Coman, M.A. Nascimento, and J. Sander. A framework for spatio-temporal query processing over wireless sensor networks. In *Proc. of DMSN Workshop*, pages 104–110, 2004.

[5] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proc. of ICDE*, pages 449–460, 2004.

[6] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing historical information in sensor networks. In *Proc. of SIGMOD*, pages 527–538, 2004.

[7] A. Demers, J. Gehrke, R. Rajaraman, N. Trigoni, and Y. Yao. Energy-efficient data management for sensor networks. In *Proc. of Upstate New York Workshop on Sensor Networks*, 2003.

[8] A. Deshpande, C. Guestrin, W. Hong, and S. Madden. Exploiting correlated attributes in acquisitional query processing. In *Proc. of ICDE*, 2005 (to appear).

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

[10] A. Deshpande, S. Nath, P.B. Gibbons, and S. Seshan. Cache-and-query for wide area sensor databases. In *Proc. of SIGMOD*, pages 503–514, 2003.

[11] Q. Fang, F. Zhao, and L. Guibas. Counting targets: Building and managing aggregates in wireless sensor networks. Technical Report P2002-10298, PARC, 2002.

[12] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. In *Proc. of SenSys*, 2003.

[13] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden. Distributed regression: an efficient framework for modeling sensor network data. In *Proc. of IPSN*, pages 1–10, 2004.

[14] W. Heinzelman. *Application-Specific Protocol Architectures for Wireless Networks*. PhD thesis, MIT, 2000.

[15] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE Trans. on Networking*, 11(1):2–16, 2003.

[16] B. Karp and H.T. Kung. Greedy perimeter stateless routing for wireless networks. In *Proc. of MobiCom*, pages 243–254, 2000.

[17] S. Madden and M.J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proc. of ICDE*, pages 555–566, 2002.

[18] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proc. of SIGMOD*, pages 491–502, 2003.

[19] V. Raghunathan, C. Shurgers, S. Park, and M.B. Srivastava. Energy aware wireless microsensor networks. *Signal Processing Magazine*, 45(2):40–50, 2002.

[20] T. Rappaport. *Wireless Communications: Principles and Practice*. Prentice-Hall Inc., 1996.

[21] S. Ratnasnmy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A geographic hash table for data-centric storage. In *Proc. of WSNA Workshop*, 2002.

[22] A. Ricadela. Sensors everywhere. *Information Week*, Jan. 24, 2005.

[23] M.A. Sharaf, J. Beaver, A. Labrinidis, and P.K. Chrysanthis. Tina: A scheme for temporal coherency-aware in-network aggregation. In *Proc. of MobiDE Workshop*, pages 69–76, 2003.

[24] E. Shih, S.H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *Proc. of MobiCom*, pages 272–287, 2001.

[25] I. Stojmenovic. Position based routing in ad hoc networks. *IEEE Communications Magazine*, 40(7):128–134, 2002.

[26] Y. Yao and J. Gehrke. Query processing in sensor networks. In *Proc. of CIDR*, 2003.

[27] D. Zhang, D. Gunopulos, V.J. Tsotras, and B. Seeger. Temporal and spatio-temporal aggregations over data streams using multiple time granularities. *Information Systems*, 28:61–84, 2003.