Retrieval of Spatial Join Pattern Instances from Sensor Networks*

Man Lung Yiu
Department of Computer Science
Aalborg University
DK-9220 Aalborg, Denmark
mly@cs.aau.dk

Nikos Mamoulis
Department of Computer Science
University of Hong Kong
Pokfulam Road, Hong Kong
nikos@cs.hku.hk

Spiridon Bakiras
Dept. of Mathematics and Comp. Science
John Jay College
City University of New York
sbakiras@jjay.cuny.edu

Abstract

We study the continuous evaluation of spatial join queries and extensions thereof, defined by interesting combinations of sensor readings (events) that co-occur in a spatial neighborhood. An example of such a pattern is "a high temperature reading in the vicinity of at least four high-pressure readings". We devise acquisitional and distributed protocols for evaluating this class of queries, aiming at the minimization of energy consumption. Cases of simple and complex join queries with single or multi-hop distance constraints are considered. Finally, we experimentally compare the effectiveness of the proposed solutions on an experimental platform that simulates real sensor networks. Our results show that acquisitional protocols perform best for multi-hop or high-selectivity queries while distributed techniques should be applied for the remaining cases.

1 Introduction

Advances in computer hardware have brought to availability small and relatively cheap devices forming a powerful network that interacts and collects information from the environment, where it is deployed. Sensor networks have several applications, including environmental monitoring [14, 12] and control/maintenance of industrial infrastructure [1]. Recently, the problem of evaluating queries over a sensor network has attracted significant research interest from the database community, leading to the development of two research DBMS prototypes [18, 13]. These systems provide to the user an interface, via which queries are expressed in a *declarative* way (e.g., SQL extensions); the user needs not deal with *how* queries are evaluated.

The main focus of existing work on sensor networks has been the minimization of power consumption at sensor nodes, during query evaluation. Sensors are usually battery-operated and they are often deployed in hostile environments or rough terrains, where the network runs unsupervised for long time intervals. Thus, power is of utmost importance, since it is directly related to the longevity of the network. Previously studied topics include the energy-efficient retrieval of aggregations or data summaries [12, 5, 3, 7, 6, 16], the derivation and maintenance of data models that describe the data distribution [8, 4], and the optimal in-network placement of operators or filter predicates on the sensed values [13, 2, 1, 17]. To our knowledge, we are the first to study in-network evaluation of queries that *spatially correlate* measurements from multiple, different sensors. An example of such a query is "generate a notification whenever a sensor with high temperature reading is 10 yards from four sensors with low humidity readings". A *spatial pattern query* retrieves sets of sensors (pairs in this example), whose readings qualify some selection predicates (e.g., abnormal temperatures) and their locations qualify some pairwise distance predicates (e.g., within five yards). Data analysts may be interested in the on-line identification of pattern instances that occur rarely in the environments where sensors are deployed and may indicate exceptional events. For instance, an unusually high temperature detected in the vicinity of multiple low-humidity readings may indicate high chance of a fire break in the local area, where the pattern is detected. Another application of spatial pattern queries is the prediction of weather phenomena based on spatial combinations of sensor readings.

A straightforward way to evaluate spatial pattern queries is to program the sensors to transmit their readings together with their locations to a central basestation (via a routing tree [9, 13]), where their spatial associations are validated. However, this approach may waste more energy than necessary, as sensor readings that are not part of query results may be sent all the way up to the root. Motivated by the lack of effective evaluation protocols for spatial pattern queries, in this paper, we study this problem in depth, focusing on (i) filtering techniques for readings that do not participate in the result, (ii) in-network computation of query results. We propose optimized evaluation protocols for binary spatial joins and more complex query patterns and compare them for different problem parameters. Our solutions are orthogonal

*Work supported by grant HKU 7155/06E from Hong Kong RGC.

to snapshot-based schemes (e.g., [10]), which apply query evaluation only to a small (self-maintained) sample of the network and to techniques that summarize sensor readings over long time intervals before applying query evaluation on them (e.g., [6]). The contributions of this paper can be summarized as follows:

- We identify the interesting class of *spatial pattern queries*. We formally define them and express them using the language extensions of [13].
- We propose energy-efficient protocols for in-network evaluation of spatial pattern queries, based on both acquisitional and distributed evaluation.
- We experimentally evaluate the efficiency of the proposed techniques with various parameters, e.g., query selectivity, network size, topology, sampling cost, etc.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 formally defines spatial pattern queries. In Section 4, we describe in detail the proposed solutions. Section 5 experimentally demonstrates the efficiency of our techniques. Finally, Section 6 concludes the paper.

2 Background and Related Work

The special characteristics of a sensor network compared to a generic wireless network are (i) the limited resources of nodes (energy, communication range, network bandwidth and capacity), (ii) unreliable communication with high packet loss rates and frequent node failures, and (iii) unsupervised nature with nodes placed at hostile environments (e.g., remote areas, war fields, etc.). Thus, query evaluation techniques for sensor networks aim at minimizing the energy cost, subject to the constraints of the network (e.g., communication range, maximum data volume that can be sent by a node at a cycle, etc.). Besides, sensor networks are inherently redundant (i.e., dense), in order to keep the network connected after node failures and increase the reliability of sensed information.

Query evaluation in sensor networks is performed in two steps [9, 18, 13]. Suppose that the query should collect the readings from all sensors. The query is registered at a basestation, which is connected to a *root* node r. In the first step, the query is disseminated to the sensors, and a spanning tree of the network, rooted at r is dynamically constructed. If a node receives the query for the first time, it selects one of the senders as its parent in the tree and broadcasts the query. Otherwise, the message is ignored. The resulting *communication* (or *routing*) tree is used to acquire sensor readings related to the query, up to the basestation. Delivery of sensor readings (or query results) to the root is performed in multiple phases. During a specific phase, a level of the tree *sends* and the level above listens and *receives* information addressed for it. Finally, the root collects all readings and sends them to the basestation.

Queries over sensor networks are usually *continuous*, i.e., they remain active for a lengthy time interval (e.g., minutes, hours). Otherwise, the cost for disseminating the query may not be compensated. Frequent instantaneous queries are best processed if the network operates in a push-based manner; sensors periodically and unconditionally collect measurements and route them to a basestation, where queries are registered and evaluated as queries over streaming data. In this paper, we study continuous queries, which can benefit from in-network evaluation. Next, we review work on (continuous) query evaluation on sensor networks.

2.1 Aggregation and summarization

Madden et al. [12] proposed a simple, but powerful protocol for computing common aggregate functions (e.g., count, sum, max, min). Each sensor combines the information received by its children with its own measurement to derive and send data of constant size, capturing a partial computation of the aggregate function. In [5], a multi-path algorithm for computing aggregates is presented to reduce communication errors as multiple parents may hear and aggregate the information broadcast by a single child. [15] proposes a hybrid method that combines the tree topology of [12] with the ring network topology of [5]. Besides, [7] describes a method for pushing error tolerance in network nodes, in order to avoid sending information if the aggregate is within some error bound. The problem of redistributing the error tolerance among nodes in order to minimize the overall error at dynamic environments is also studied. A similar approach was independently proposed in [16]. To minimize network communication, [6] presents a methodology for in-network compression of multiple (time-series) signals generated by sensors (e.g., one for temperature, one for humidity, etc.). The rationale is that measurements observed at the same node are likely to follow similar trends.

2.2 Data models, snapshots, and filters

An alternative to continuously collecting and processing sensor data (which drains the network energy resources), is to define and maintain simple data models (e.g., mixtures of Gaussians) for the data distribution [8, 4]. These models, potentially combined with exact readings, provide query answers with some approximation confidence. Besides, [10] describes a framework for dynamically selecting and maintaining representatives in a redundant sensor network. The set of representatives (snapshot) plays the role of a dynamic sample that can answer queries cheaply and approximately.

Another class of problems is the distribution of filters or database operators in the routing tree of a sensor network. [17] studies the optimal placement of query operators (e.g., selection predicates), in order to minimize (i) the communication cost for information that does not end up in the query result and (ii) the computational burden at lower tree levels (assuming that lower-level nodes have reduced computational capabilities). [2] focuses on the assignment of operators that correlate measurements from two (apriori defined) spatial regions. [20] examines a similar problem and applies synopses of sensor values to eliminate unqualified readings that cannot lead to results. On the other hand, our problem searches for rare spatial associations of (instantaneous) events, anywhere in the network map. In another direction, [1] studies continuous joining a table of predicates (e.g., 'humidity>50^{o}C') with the sensed values. If the table is small enough to be stored at each node, it acts a filter that prevents non-qualifying readings to be sent to the basestation. If the table cannot fit in a node's local memory, it is placed at neighboring nodes and the predicates are evaluated in a distributed fashion. However, the queries we study do not simply consider sensor values; they also have to satisfy a spatial pattern, which will be defined formally in the next section.

The closest work to ours is [11], which reports pairs of sensor events located within a given distance range, and reduces communication cost by a distributed routing index. The sensors record past events in their neighborhood which help to predict future occurrences of them at other locations of the map. Messages are then routed based on these predictions. As the author suggests, the index is appropriate for applications where events correspond to moving objects with well-estimated future locations. Our focus, on the other hand is on arbitrary, instantaneous, ad-hoc events. In addition, the methodology of [11] relies heavily on the regular grid networks and may not be applicable to arbitrary network topologies.

3 Problem Formulation

Let $\mathcal{SN}$ be a network of N sensors. Each sensor $s \in \mathcal{SN}$ is associated with a spatial *location*[1] $s.loc$, and can produce a set $s.m$ of measurements (e.g., temperature, humidity, etc.) for the spatial region around it (different sensors might produce different sets of readings, in general).

We adopt the framework described in Section 2, where users register continuous queries at a basestation and a routing tree is created to acquire results (or readings that are processed at the base). Each registered query is associated with: (i) a *lifetime* (e.g., 2 hours), during which it is active and continuously produces results from the sampled measurements, and (ii) an *epoch duration* (e.g., 10 seconds), every which the network samples measurements. In other words, queries apply to instances of the network at different timestamps (for every epoch).

A binary *spatial pattern* query identifies pairs of sensors, for which (i) the readings qualify some particular selection predicates and (ii) the locations are no further than a particular distance from each other. An example of such a query is "find pairs of sensors $\langle s_1, s_2 \rangle$, such that s_1.temperature>50^{o}C, s_2.humidity<40%, and $distance(s_1, s_2)$<10m". A generalized spatial pattern query (formally defined below) returns sets of sensors, whose values and locations qualify some selection predicates and binary distance predicates, respectively.

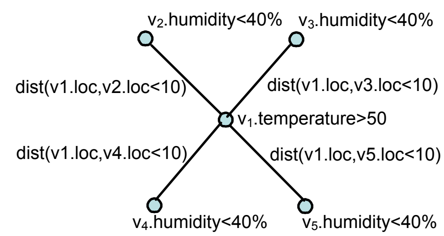


Figure 1. A spatial pattern query

Definition 1 *A spatial pattern query Q consists of a set $Q.\mathcal{V}$ of variables, a set $Q.\mathcal{P}$ of selection predicates, and a set $Q.\mathcal{B}$ of binary constraints. Each variable $v_i \in \mathcal{V}$ is associated to a selection predicate P_i. A pair $\langle v_i, v_j \rangle$ of variables $v_i, v_j \in \mathcal{V}, i \neq j$ may be associated with a binary spatial predicate B_{ij}. A result of Q is set of assignments $\{\forall v_i \in \mathcal{V} : v_i \leftarrow s_i, s_i \in \mathcal{SN}\}$, such that (i) for all v_i, $P_i(v_i.m)$ is satisfied and (ii) for all variable pairs (v_i, v_j) with a binary predicate, $B_{ij}(v_i.loc, v_j.loc)$ is satisfied.*

Figure 1 shows an example of a spatial pattern query q modeled by a graph. Each node in the graph corresponds to a variable, whose values are constrained by a selection predicate. Edges correspond to binary spatial predicates (i.e., distance constraints). In natural language, q could be expressed as "a high-temperature reading (>50^{o}C) in the vicinity of four low-humidity readings (<40%)". A local group of sensors whose readings satisfy this query could indicate an area that requires special attention (e.g., high chances of fire, if a forest).

Spatial pattern queries can be easily expressed in the extended SQL of [13], assuming that the language supports spatial functions (i.e., distance). The query variables (defined in the `FROM` clause) are instantiated by tuples of the Sensors table and the selection while join predicates (i.e., distance constraints) are connected by `AND` in the `WHERE` clause. Although Definition 1 is generic enough to define queries of arbitrary graphs and constraints, we confine our attention mainly to binary joins and to extended patterns that form "star" graphs (like the one in Figure 1), where a centric feature (e.g., high temperature) is correlated to a number of other features (e.g., low humidity) in its surrounding environment. Such patterns were shown important in spatial analysis applications and are more intuitive than

[1] We assume that the locations of sensors are known to them. They could be constant and apriori defined (for stationary, manually placed sensors), or detected by GPS devices placed on the sensors.

queries that combine variables in an arbitrary graph. The centric feature models a point of interest (e.g., high fire risk area, expensive equipment) which should trigger an alert whenever its local measurements and the conditions in the region around it form an abnormal combination.

4 Proposed Methods

In this section, we explore the applicability of several methods for computing spatial pattern queries in a sensor network. We divide the evaluating protocols in two classes. The class of *acquisitional* protocols collect sensor measurements via the communication tree and apply query evaluation at the basestation. Filters are placed at nodes that generate or relay data to minimize the transferred volume. The second class of *distributed* protocols apply in-network query evaluation and send the results to the basestation (again using the tree). We start by discussing the simple case of a binary spatial join with distance constraint smaller than the communication range of the nodes. Then, we extend the suggested protocols for more complex queries and multi-hop distance constraints.

4.1 Single-hop binary joins

We first focus on binary join patterns that are sensor pairs $\langle s_i, s_j \rangle$, such that $P_1(s_i.m)$, $P_2(s_j.m)$ are satisfied, and $distance(s_i.loc, s_j.loc) \leq c$, where c is smaller than the radio communication range[2] between two nodes. For the ease of exposition, we denote a binary join query in our context by the triplet $\langle P_1, P_2, c \rangle$.

4.1.1 Brute-force acquisitional protocol

The straightforward way to evaluate the query is to program all sensors to sense the measurements relative to selection predicates P_1 and P_2, at every epoch, and send this information to the basestation, which evaluates the spatial join locally. A simple optimization that reduces the number of unnecessary values transmitted to the base is to "push-down" the selection predicates at the nodes (as suggested in [13, 1]). In our example, temperature and humidity are sensed by all sensors but only high temperature and low humidity values (i.e., those that qualify the selection predicates) are sent to the base. In order to minimize the transferred data, we only transmit the location of a qualifying node (or its identifier, if nodes have fixed locations) and two bits that indicate which predicate(s) the node qualifies (e.g., 10 implies that P_1 is qualified, but P_2 is not). Sensors are synchronized such that only two consecutive levels of the tree are active at the same phase (while the remaining nodes are sleeping), as discussed in Section 2. When a lower-level node senses and transmits data (if not filtered by P_1 or P_2) to its parent, its parent listens, reads and combines its readings with those of its children; the combined readings are then sent to the upper level during the next phase. We denote this simple, but generic protocol by AQB (i.e., the first 'acquisitional' protocol).

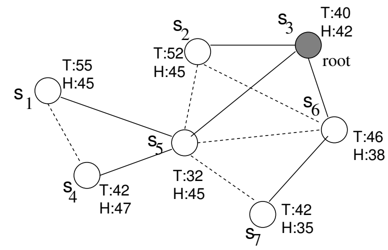


Figure 2. Join evaluation example

As an example, consider the sensor network depicted in Figure 2. Nodes within communication range from each other are connected by edges. Solid edges denote the structure of the communication tree (rooted at node s_3). The values next to the nodes denote the (current) local temperature (T) and humidity (H) conditions. Let P_1='T>50', P_2='H<40', and c equals the sensor communication range (one hop). Nodes s_1 and s_2 qualify P_1, whereas s_6 and s_7 qualify P_2. The only join result is $\langle s_2, s_6 \rangle$. In the first phase of the cycle, s_1, s_4, and s_7 (level-3 nodes) sense their values, apply the predicates and s_1 sends $(s_1.loc, 10)$ (i.e., only P_1 is satisfied) to its parent (i.e., s_5). Similarly, $(s_7.loc, 01)$ is sent to s_6. In the second phase, $(s_2.loc, 10)$, $(s_1.loc, 10)$, and $\{(s_7.loc, 01), (s_6.loc, 01)\}$ are sent to the root, by s_2, s_5, and s_6, respectively. Finally, s_3 forwards all these tuples to the base, where the join result is computed.

4.1.2 Pruner-based acquisitional protocol

Protocol AQB may send more information than necessary to the base, as many tuples (e.g., $(s_1.loc, 10)$ in Figure 2) are likely not to participate in the spatial join. In this section, we propose AQP, a protocol that improves upon AQB, by adding more sophisticated filters in the intermediate nodes of the tree. AQP is based on the observation that a sensed value (e.g., m) of a node s_i which satisfies a selection predicate (e.g., P_1) can be pruned by an ancestor $a(s_i)$ of s_i, if (i) $a(s_i)$ collects all information about the spatial neighborhood of s_i and (ii) no matching tuple (e.g., one that qualifies P_2) for the measurement has been collected by $a(s_i)$. For example, $(s_1.loc, 10)$ in Figure 2 can be pruned by s_5, since any measurements that qualify P_2 within one hop from s_1 should have been collected or generated by s_5.

Formally, let $\langle P_1, P_2, c \rangle$ be a join query registered over a sensor network $\mathcal{SN}$. For each sensor $s \in \mathcal{SN}$, we define its *neighborhood sensor set* $L(s)$ as $L(s) = \{s' \in \mathcal{SN} | dist(s, s') \leq c\}$, and its *descendant sensor set* $B(s)$ as the set of sensors in its subtree (of the routing tree). The

[2] Without loss of generality, we assume that all sensors have the same communication range. Our protocols and filtering techniques can be easily adjusted for the generic case.

pruning technique applied in AQP is based on the Lemma 1 (with trivial proof):

Lemma 1 *Let s_i be a sensor satisfying $P_1(s_i.m)$. Let $a(s_i)$ be an ancestor of s_i, such that $L(s_i) \subseteq B(a(s_i))$. If there is no $s_j \in B(a(s_i))$ satisfying $P_2(s_j.m)$, then $s_i.m$ cannot participate in an output tuple of $\langle P_1, P_2, c\rangle$. A symmetric argument holds for the measurements which qualify P_2.*

For any sensor s, there is at least one ancestor (the root) for which $L(s) \subseteq B(a(s))$, thus we can apply this idea to prune measurements acquired from the network that do not participate in query results. The goal is to find the closest ancestor of s to apply the filter, since, in this way, filtering effectiveness is maximized. For each node s, the *pruner* of s (with respect to a query) is the nearest ancestor of s, whose subtree contains $L(s)$. In Figure 2, s_5 is the pruner of s_1 and s_4.

We design the following technique for determining pruners efficiently. It is applied only once while the query is disseminated and the routing tree is constructed. When a sensor node s broadcasts the query, at the same time it collects ids/locations of its neighbors and determines $|L(s)|$ (the cardinality of $L(s)$), which is then broadcasted to all nodes in $L(s)$. Starting from the leaf nodes, each sensor s sends up the communication tree a table consisting of $\langle s_i, |L(s_i)|, 1\rangle$ tuples for all $s_i \in L(s)$ plus a $\langle s, |L(s)|, 1\rangle$ tuple for itself. Intermediate tree nodes merge the tables they receive from their children by summing their counters (the last field of the tuples). The first node which, after the aggregation, has a $\langle s, |L(s)|, |L(s)|\rangle$ tuple becomes the pruner for s and does not forward the tuple to its parent node.

After this process, each node s keeps a list of its *prunees* (i.e., nodes for which s is the pruner). The difference between our improved acquisitional protocol AQP and the baseline AQB is that, in AQP, whenever s retrieves information from its subtree regarding the join query, for each prunee that transmits a tuple τ qualifying P_1 (P_2), s checks whether there is a matching tuple for P_2 (P_1) in its acquired table, which also qualifies the distance predicate with τ. If there is no such tuple, then τ is pruned from the data sent to the parent of s. AQP manages to filter early some node readings (e.g., $(s_1.loc, 10)$) that do not qualify the join predicate.

4.1.3 Distributed evaluation

The class of *distributed* protocols aim at computing query results locally around network nodes and sending them to the basestation. Such a technique is expected to pay-off for low-selectivity[3] joins, where many measurements that satisfy predicates P_1 or P_2 do not qualify the join condition. During the first stage of distributed evaluation, nodes that qualify P_1 and P_2 communicate and determine the join results. During the second stage, the routing tree is used to send the join results to the base.

Initially, all nodes sense the measurements related to P_1 and P_2. If a node s_i qualifies P_1, it broadcasts its location to its neighborhood. If a node s_j qualifies P_2, it listens for potential messages from nodes that qualify P_1. For each received message, s_j produces a join result. Nodes that qualify neither P_1 nor P_2 remain asleep until the first stage terminates (they may have to wake and forward join results, during the second stage). Note that the roles of P_1 and P_2 could be interchanged; we denote by DS1 (DS2) the distributed protocol, where nodes qualifying P_1 (P_2) send messages and those qualifying P_2 (P_1) receive them and compute join results. Intuitively, DS1 should be preferred to DS2 when nodes that qualify P_1 are fewer than those qualifying P_2, since transmission is more expensive than listening and receiving [14].

As an example, consider again the network of Figure 2. In the first stage of the distributed protocol, measurements are collected, and (i) nodes s_1 and s_2 (qualifying P_1) broadcast their locations, (ii) nodes s_6 and s_7 (qualifying P_2) listen for potential messages, (iii) nodes s_3, s_4, and s_5 stay asleep. After node s_6 reads the transmission of s_2, the join result $\langle s_2, s_6\rangle$ is formulated. This is the only tuple that will be forwarded to the root at the second stage (result acquisition).

So far we have ignored the cost for sensing measurements at nodes, which is usually small compared to communication costs. For some measurements, however, this cost may be significant [13]. For cases where sensing for P_2 is significantly expensive, it might be beneficial to defer sensing and instruct all nodes to listen for P_1 messages. Only if a listener receives a message from a P_1 node, it performs expensive sensing for P_2 measurements. We denote this protocol by DS1′.

4.2 Complex join queries

We now consider more complex pattern queries, as described in Section 3. Queries correspond to star graphs, where the center sensor node should satisfy selection predicate P_C and there are k border nodes that should qualify P_B within distance c from the center. A star pattern query is simply denoted by a quadruple $\langle P_C, P_B, c, k\rangle$. As in Section 4.1, we assume that c is at most equal to the radio range of the sensors.

Acquisitional protocols We can directly apply the brute-force acquisitional protocol AQB. Sensor readings that qualify P_C or P_B are unconditionally sent to the basestation, where the pattern is evaluated. In addition, we can adapt protocol AQP as follows. A tuple qualifying P_C which has been generated by a node s_i is filtered at node $pr(s_i)$ (pruner of s_i) if there are less than k tuples that qual-

[3] Low-selectivity joins output few results while high-selectivity joins produce many results.

ify P_B and reach $pr(s_i)$ (otherwise, we know that there may be a query result that contains the tuple). A tuple qualifying P_B, which has been generated by a node s_i is filtered at $pr(s_i)$ if there is no tuple satisfying P_C that reaches $pr(s_i)$ (i.e., similar to binary join queries).

Distributed protocols A simple way to extend the distributed protocols for complex queries is to ask 'border' nodes (those qualifying P_B) broadcast their locations. At the same time 'centric' nodes (those qualifying P_C) listen for potential messages. If a centric node s_i, receives at least k messages, we know that there is a query result centered at s_i.[4] The query result is sent to the base station through the routing tree, at the second stage of the protocol. We denote this protocol by DSB. An alternative protocol aims at minimizing the messages broadcast from border nodes; presumably more sensors qualify P_B than P_C for the pattern query to have small selectivity and correspond to an interesting, exceptional event. Protocol DSC asks nodes that qualify P_C (center nodes) to send a message and nodes that qualify P_B (border nodes) to listen. If a border node receives a message it sends a response with its location to its neighbors. Finally, center nodes listen for messages and those that hear from at least k nodes send the query result to the base.

4.3 Multi-hop queries

Distance constraints longer than the radio range h impose difficulties for distributed evaluation protocols. Given a node s, there is no bound for the number of hops required to find the nodes within distance c ($>h$) from s. Nonetheless, for a relatively dense and uniform network, we could set an approximate upper bound λ for this number. Let *coverage*(c,λ) be the probability that two sensor nodes within distance c are reachable within λ hops. Figure 3 plots the coverage as a function of λ on a typical random network (with the default parameter values discussed in Section 5). For instance, for the curve of "$c = 3h$", c is set to 3 times of the radio range h. Observe that the coverage increases rapidly when λ increases. In order to balance the coverage and energy consumption, we suggest to set $\lambda = \left\lceil \frac{c\sqrt{2}}{h} \right\rceil$ for multi-hop communication. We now discuss in more detail the protocols that can be applied for queries that involve multi-hop distances.

Acquisitional protocols Since AQB does not apply any filtering or in-network evaluation, there is no difference than the method described in Section 4.1 for multi-hop queries. For AQP, the only difference is in the initialization of the query, at the stage when pruners are defined. Each node

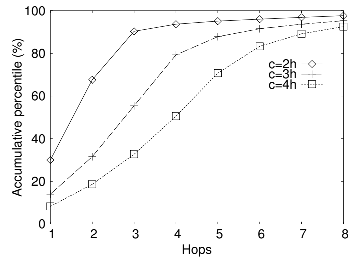


Figure 3. Coverage in multi-hop communication

needs to determine the number of its λ-hop neighbors before sending it up the communication tree. This process requires flooding a large number of messages and it is more expensive than the simple 1-hop communication. However, it is performed only once, during the initialization of the routing tree and it is expected to pay off if the query has long lifetime.

Distributed protocols The distributed protocols described so far can be easily adapted for multi-hop queries, at the expense of higher communication cost, since the whole network may need to stay up in order to listen and relay potential messages, during the first stage (computation of query results). If λ is large, the cost of flooding may be too high for distributed evaluation to pay-off. In such cases, the acquisitional protocols are expected to dominate.

A bi-directional distributed protocol For queries that are simple binary joins, we can apply a *bi-directional* distributed protocol (BD) in order to reduce message flooding during the first (computation) stage. Instead of asking nodes that qualify P_1 to flood their locations up to λ hops (which are then received by listeners that qualify P_2 and converted to query results), we ask all nodes that qualify either P_1 or P_2 to send their locations and a pair of bits indicating the qualified predicates (i.e., the information sent by nodes to the base according to AQB/AQP). However, the flooding range is now reduced. Nodes that qualify P_1 send their messages up to x hops ($x < \lambda$) and nodes that qualify P_2 up to $\lambda - x$ hops.[5] During this process all nodes of the network are up in order to listen and relay messages. If a node receives a message from both a P_1 node and a P_2 node, it formulates and caches the join result. In the second stage of the algorithm, nodes send the computed results up the tree to the basestation. Note that duplicate results could be computed, since the same pair of messages may be received by the same node. For instance, consider a query that seeks for high-temperature/low-humidity readings within $\lambda = 2$ hops in the network of Figure 2. When

[4] In fact, if s_i receives $m > k$ messages, we have multiple query results, one for each $\binom{m}{k}$ combination of border nodes. Nonetheless all these results can be compressed to a single tuple containing s_i and all qualifying border nodes.

[5] A node that qualifies both predicates sends its message up to $\max\{x, \lambda - x\}$ hops.

BD is applied, all nodes that sense either high temperature (>50) or low humidity (<40) transmit their readings up to 1 hop (i.e., $x = 1, \lambda - x = 1$). Both s_5 and s_6 then identify $\langle s_2, s_7 \rangle$) as a result. Duplicate results are eliminated by merging operations at the second stage of the protocol, when all results are sent up the communication tree.

An interesting problem is to pick a value of x such that the communication cost is minimized. In general, x can take $\lambda + 1$ values (the extreme cases $x = 0, x = \lambda$ correspond to the uni-directional distributed protocols DS1 and DS2). Intuitively, x should be chosen to minimize the expected expansion area $\pi(Sel(P_1) \cdot x^2 + Sel(P_2) \cdot (\lambda - x)^2)$, where $Sel(P_i)$ corresponds to the probability that a node qualifies P_i (for $i = \{1, 2\}$). In other words, if P_1 has low selectivity (few nodes qualify it) compared to P_2, nodes that qualify it should transmit far and nodes that qualify P_2 should transmit close in order to minimize network traffic.

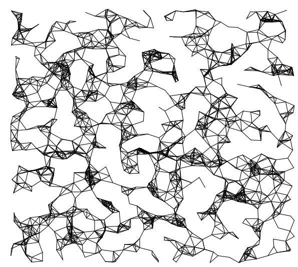
5 Experimental Evaluation

In this section, we evaluate the efficiency of the proposed protocols on an experimental platform that simulates real sensor networks. Table 1 shows the components we consider when measuring query cost (taken from [14]). In all (but one) experiments, the selection predicates are applied on the cheapest to sense measurements, thus the sensing cost is negligible compared to communication/listening costs. We do not count the computational cost, since the operations involved in our protocols are cheap filters or distance checks. The packet size (excluding the header) was set to 30 bytes (typical for MICA motes [12]). Our protocols pack multiple events/messages in one packet, before transmitting them. The acquisitional protocols use 18-bit messages (node-id or coordinates plus 2 bits for indicating qualified predicates). The distributed protocols use 32-bit messages for sending pairs of node ids/coordinates to the root. As in [1], we assume long-running queries and do not count the one-time cost of initializing the query in the network. In each experiment, we run a protocol for 100 epochs and record the average cost per epoch.

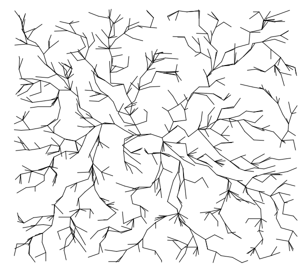
Operation	Cost (nAh)
Transmitting a packet	20
Receiving a packet	8
Idle listening (for 1 ms)	1.25
Thermistor sample	0.35
Barometric pressure sample	1.39
Photoresistor sample	3.43
Infrared sample	9.44
I^2*CTemperature sample*	20.83

Table 1. Costs of MICA operations

We experimented with a random (uniform) network topology. Experiments with other network topologies can be found in our technical report [19], which also contains an analytical study for the performance of the protocols (omitted due to space constraints). The default network size is $N = 1024$ nodes. To generate the RANDOM network, we randomly placed nodes inside a square area of side $\sqrt{N}$ and set the radio range to 1.5. These settings result in a network that is fully connected and not extremely dense, as shown in Figure 4. In Figure 4a, nodes within one hop are connected by line segments. Figure 4b shows the corresponding routing tree, where the base station is located at the center of the space. The average degree of a sensor node is 6.8. The root node of the routing/aggregation tree is chosen as the center of the network [12, 5] and the tree height is 24. Unless otherwise stated, the selectivities of unary predicates (i.e., P_1, P_2) are set to 0.05. For a single-hop binary join, these settings return 20 join results on average.



(a) network



(b) routing tree

Figure 4. RANDOM topology

5.1 Single-hop binary joins

We first study the performance of the proposed protocols for low-selectivity single-hop binary join queries (with two selection predicates P_1 and P_2). Protocols AQB, AQP, and the distributed protocol (described in Section 4.1) are compared. By default, the selectivities of P_1 and P_2 are equal, so DS1 is equivalent to DS2; we simply denote either of them by DS.

Figures 5 shows the averaged costs (with error bars) of the three protocols as a function of the join selectivity. The join output size was controlled by tuning $Sel(P_1)$ ($=Sel(P_2)$). For joins with few results, Protocol DS is more efficient than AQB and AQP because pruner nodes (in AQP) are located several levels above their prunees and measurements that qualify the selections participate in very few or no join results. As the join output size increases, the energy consumption increases for all protocols, as more tuples are transferred to the base, but the relative performance of DS compared to AQB/AQP decreases, as the number of join results compared to the tuples that qualify either P_1 or P_2 increases. Eventually, DS becomes worse than the acquisitional protocols, since all readings that qualify the selections participate join result and the join output size well exceeds number of tuples that qualify either selection.

Figure 6a provides statistics about the effectiveness of

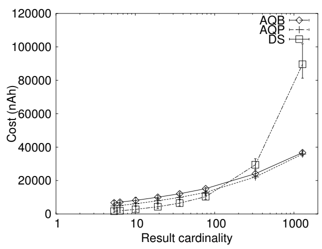


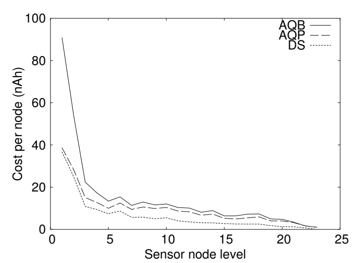
Figure 5. Effect of join output size

pruner nodes in AQP. The table distributes the 1024 nodes of each network into classes based on the percentage of hops saved if their tuples are pruned by AQP. For instance, if a node s falls into the $80\%-100\%$ class, then the quantity $\frac{\text{hops between } pr(s) \text{ and the base}}{\text{hops between } s \text{ and the base}}$ (i.e., the path ratio saved if a tuple from s was pruned by $pr(s)$) is between 0.8 and 1. Since most of the pruners have high hops-saving ratio, the overall pruning effectiveness is quite high.

Next, we verify the assertion that AQP and DS achieve better cost balancing than AQB among different nodes. Figure 6b shows the average cost per node as a function of node's level in the routing tree. In general, sensor nodes at higher levels receive and forward more data so they have larger burden. DS and AQP have better balancing, since they manage to eliminate tuples that do not participate in join results early, either by computation of the exact join results (DS) or by filtering tuples at pruner nodes (AQP).

Ratio(%)	Pruners
80-100	414
60-80	250
40-60	165
20-40	105
0-20	90

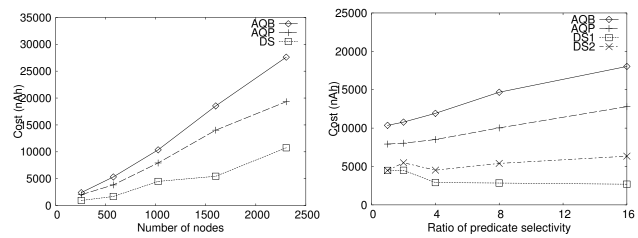
(a) Hops-saving ratio (in AQP)



Cost balancing

Figure 6. Hops-saving ratio, cost balancing

In the next set of experiments, we test the performance of the protocols. Figure 7a shows the cost of the protocols as a function of number of nodes, while keeping the network density fixed. Note that the cost difference between the protocols is not greatly affected by the network size. So far, we have assumed that P_1 and P_2 have the same selectivity. We now test the effect of unbalanced selectivities at the selection predicates (Figure 7b). For this experiment, we kept the product of the two selectivities fixed and varied the ratio $r = Sel(P_2)/Sel(P_1)$. For various values of r we plot the energy consumption by the different protocols. Since $Sel(P_1) \neq Sel(P_2)$, we split protocol DS to DS1 and DS2. DS1 is more efficient than DS2 for $r > 1$ and its cost decreases with r. As r increases, the number of sensors that qualify P_1 decreases, and so do the transmitted messages by DS1. Although the number of listeners (i.e., nodes that qualify P_2) increases, the listening (and reading) cost is significantly lower than the transmission cost (see Table 1), thus the overall cost of DS1 drops. On the other hand, the cost of DS2 (slightly) increases with r, due to the increased number of transmissions. Acquisitional protocols get more expensive with r, since the sensors that qualify either P_1 or P_2 increase.[6]



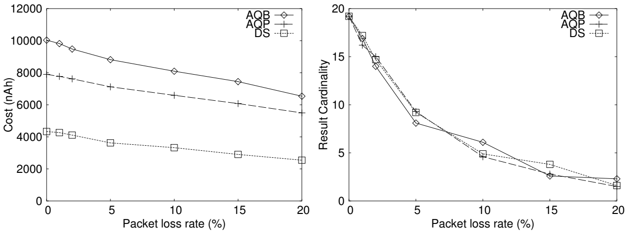
(a) scalability (b) effect of $\frac{Sel(P_2)}{Sel(P_1)}$

Figure 7. Network size, predicate skew

A natural question for advanced sensor network protocols is whether any additional operations performed by them affect the data loss rate, due to communication errors. We first evaluate the effect the packet loss rate has on the performance of the algorithms (Figure 8a). As the plot shows, the relative performance of the methods is not affected by this factor. Figure 8b shows the join output size as a function of packet loss rate. Observe that similar number of results are detected by different protocols. Thus, the functionality of the protocols does not affect the result loss rate in lossy networks. On the other hand, even with relatively low packet loss rates (10%) a large percentage of results is not detected. This is expected, as the probability of a join result (or a component tuple in a join result) to reach the basestation decreases exponentially with the number of hops the message needs to travel. The reliability of the network during the acquisition phase can be increased, by applying multi-path routing techniques paired with duplicate elimination mechanisms (e.g., [5, 15]), instead of the routing tree [13]. Intuitively, protocol DS is more appropriate for multi-path routing than AQP, since (i) the amount of transferred data is low as only (rare) join results are routed and (ii) the pruner nodes of AQP will be less effective, since tuples from prunees may find other paths to the root.

We now examine a case where sampling data is significantly expensive. We consider a query, where P_1 applies on barometric pressure, P_2 applies on I^2C temperatures

[6] For a fixed product $Sel(P_2) \cdot Sel(P_1)$, the probability for a sensor to qualify either P_1 or P_2 (i.e., $1 - (1 - Sel(P_1))(1 - Sel(P_2))$) is minimized when $Sel(P_2) = Sel(P_1)$ and increases with r.



(a) cost (b) result cardinality

Figure 8. Effect of packet loss rate

and the selectivity of each predicate is 0.05. We compare the original protocol DS, with the variant of it, described in Section 4.1, which asks all nodes to unconditionally listen to messages from nodes that qualify P_1. Only nodes that receive messages apply sampling to verify the selection condition of P_2. We denote this protocol by DS′. Table 2 displays the cost-breakdown of the join for DS and DS′. Observe that protocol DS′ has higher packet receiving cost and idle listening cost, but it has a much lower cost on sensing the expensive measurement. In total, protocol DS′ outperforms protocol DS. In general, DS′ should be preferred to DS when (i) sampling for either P_1 or P_2 is very expensive and should not be performed unconditionally or (ii) either $Sel(P_1)$ or $Sel(P_2)$ is close to 100%; the majority of nodes qualify the predicate, so sensing should follow listening.

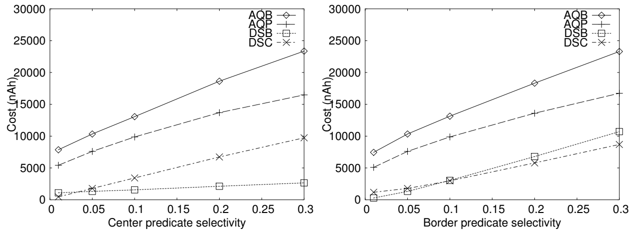
	Average nodes / epoch	
Operation	**Protocol DS**	**Protocol DS′**
Transmitting a packet	162.8	162.8
Receiving a packet	126.6	461.9
Idle listening	49.9	1024
Sensing barom. pressure	1024	1024
Sensing I^2C temp.	1024	16.2
Total cost (nAh)	27084.5	9992.0

Table 2. Cost breakdown for a query with expensive predicates

5.2 Complex joins

In this section, we evaluate the effectiveness of the protocols described in Section 4.2 for spatial pattern queries with variables forming a star graph topology.

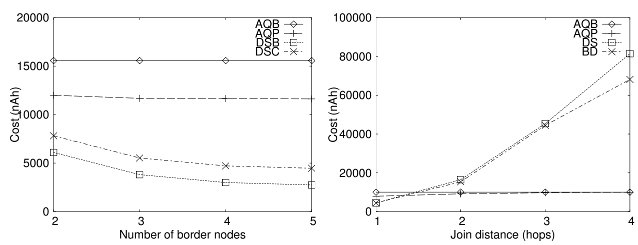
The next experiment evaluates the protocols by varying the selectivities of P_C and P_B. Figure 9a shows the cost of the protocols as a function of P_C's selectivity, with 3 border nodes and $Sel(P_B) = 0.05$. DSC has the best performance at very small values of $Sel(P_C)$. DSB starts outperforming the other protocols as $Sel(P_C)$ increases. Figure 9b shows the cost of the protocols as a function of $Sel(P_B)$, for queries with 3 border nodes and $Sel(P_C) = 0.05$. The situation is reversed in this case. DSB has the best performance at low values of $Sel(P_B)$, while DSC becomes the best protocol as the number of border nodes increases. The reason for the performance degradation of DSB is that the number of border nodes that broadcast messages is too high for unconditional transmission to pay off.



(a) varying $Sel(P_C)$ (b) varying $Sel(P_B)$

Figure 9. Effect of selectivity

Figure 10a shows the cost of the protocols as a function of number of border nodes, after fixing the selectivities of both predicates P_C and P_B to 0.1. When the number of border nodes increases, only DSB and DSC achieve significant cost reduction. For queries with many border nodes, very few results are generated and the level-off costs of DSB and DSC indicate the cost of the distributed phase. DSB is slightly cheaper than DSC, because DSC requires more nodes to transmit packets in the distributed phase.



(a) vs number of border nodes (b) vs join distance

Figure 10. Border nodes, join distance

5.3 Multi-hop queries

We now study the performance of the protocols for multi-hop binary join queries. In protocol BD, x is set to $\lambda/2$. Figure 10b plots the costs as a function of join distance, on all three network topologies. Acquisitional protocols outperform distributed protocols for join distances greater than one hop. The effectiveness of pruners remains high due to the linearity of the topology. Note that the bidirectional protocol (BD) does not have large performance difference than the purely distributed protocol. It turns out that BD has high packet reading cost, since intermediate nodes collect messages unconditionally. In addition, BD generates many duplicate join results which increase the cost of transmitting them to the basestation. In summary, acquisitional protocols are favorable for multi-hop queries, due to the extreme cost of flooding the selection results at long ranges.

Finally, we verify the trade-off of disseminating continuous queries in a sensor network and applying in-network filtering or evaluation, as opposed to continuously and unconditionally acquiring measurements, and evaluating queries at the basestation. Table 3 shows the costs of the various protocols for disseminating queries, creating the routing tree, and determining non-trivial filters (i.e., prunee information by AQP). Observe that the standard dissemination cost of the protocols (excluding prunee computation by AQP) is relatively low and can be compensated if the query runs for a long enough period (e.g, > 10 epochs), especially when $Sel(P_1)$ and $Sel(P_2)$ are small. On the other hand, the cost for computing the pruner/prunee information by AQP can be very high (especially for multi-hop queries).

	Base cost	**Extra cost by AQP**			
Hops	-	1	2	3	4
Cost (μAh)	76	181	673	1508	2767

Table 3. Cost for query dissemination

6 Conclusions

In this paper, we studied the evaluation of spatial pattern queries, which output combinations of sensor readings qualifying unary selection predicates and pairwise distance constraints. We proposed protocols that can achieve significant performance savings compared to a simple acquisitional approach that performs filtering based only on the unary selections. An improved acquisitional protocol places join filters in the routing tree that eliminate sensor readings that do not qualify the distance constraints. A distributed protocol (with variants for multi-way or multi-hop queries) performs in-network computation of the results, before sending them to the user. Experimental studies suggest that the distributed techniques perform best for low-selectivity queries with single-hop distance predicates, whereas acquisitional protocols should be preferred for multi-hop or high-selectivity queries.

References

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

[2] B. J. Bonfils and P. Bonnet. Adaptive and Decentralized Operator Placement for In-Network Query Processing. In *Proc. of IPSN*, 2003.

[3] P. Bonnet, J. Gehrke, and P. Seshadri. Towards Sensor Database Systems. In *Proc. of MDM*, 2001.

[4] D. Chu, A. Deshpande, J. Hellerstein, and W. Hong. Approximate Data Collection in Sensor Networks using Probabilistic Models. In *Proc. of ICDE*, 2006.

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

[6] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing Historical Information in Sensor Networks. In *Proc. of ACM SIGMOD*, 2004.

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

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

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proc. of MOBICOM*, 2000.

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

[11] Y. Kotidis. Processing Proximity Queries in Sensor Networks. In *International Workshop on Data Management for Sensor Networks*, 2006.

[12] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *Proc. of OSDI*, 2002.

[13] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM TODS*, 30(1):122–173, 2005.

[14] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proc. of WSNA*, 2002.

[15] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Streams. In *Proc. of ACM SIGMOD*, 2005.

[16] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. Balancing Energy Efficiency and Quality of Aggregate Data in Sensor Networks. *VLDB J.*, 13(4):384–403, 2004.

[17] U. Srivastava, K. Munagala, and J. Widom. Operator Placement for In-Network Stream Query Processing. In *Proc. of ACM PODS*, 2005.

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

[19] M. L. Yiu, N. Mamoulis, and S. Bakiras. Evaluation of Spatial Pattern Queries in Sensor Networks. Technical Report TR-2007-02, Department of Computer Science, University of Hong Kong, January 2007.

[20] H. Yu, E.-P. Lim, and J. Zhang. On In-network Synopsis Join Processing for Sensor Networks. In *MDM*, 2006.