

# Event-based Location Dependent Data Services in Mobile WSNs

Liang Hong<sup>1</sup>, Yafeng Wu<sup>2</sup>, Sang H. Son<sup>2</sup>, Yansheng Lu<sup>3</sup>

<sup>1</sup>College of Computer Science and Technology, Wuhan University, China

<sup>2</sup>Department of Computer Science, University of Virginia, USA

<sup>3</sup>College of Computer Sci. and Tech., Huazhong University of Science and Technology, China

Email: *husthong@gmail.com*, *{yw5s, son}@cs.virginia.edu*, *lys@mail.hust.edu.cn*

**Abstract**—Mobile sensors are widely deployed in Wireless Sensor Networks (WSNs) to satisfy emerging application requirements. Specifically, processing location dependent queries in mobile WSNs is still a challenging problem due to sensor mobility. We present an Event-based Location Dependent Query (ELDQ) model that continuously aggregate data in specific areas around mobile sensors of interests to provide event-based location dependent data services to the users. ELDQs generalize several typical query types and are important in many applications. However, existing approaches are incapable of efficiently answering ELDQs. In this paper, we propose a set of techniques to process ELDQs while optimizing system performance. Cost analysis and simulation results indicate that our techniques greatly reduce the cost of processing ELDQs while achieving relatively high accuracy and short response time.

## I. INTRODUCTION

Recently, mobile sensors are introduced into the Wireless Sensor Networks (WSNs) to provide various data services in many applications [15, 16, 17, 18]. However, location dependent data service queries whose results depend on the current locations of the target mobile sensors still cannot be efficiently answered by existing approaches. Specifically, we consider a class of location dependent data service queries that periodically collect and aggregate the data within the specific area around the mobile sensor nodes that detect the event. Such queries require continuous event detection as well as location dependent data aggregation on the fly, and hence need specialized optimization techniques to be processed efficiently. For example, consider succors using robots to search and rescue survivors in Sichuan earthquake. They need to query the information about the terrain and survivors around the rescue robot that detects life evidence as it moves in the unknown and dangerous environment. Base on the query results, succors can accurately locate the survivors and find the best rescue route. As another example, health providers may concern about the air pollutants ratio within a certain area around sensitive people who need preventive monitoring. The query can be “Give me the average percentage of air pollutants within 100 m around the persons whose blood oxygen levels are lower than 90% sampled every 10 minutes in the next 20 hours”. This query is triggered by the event that mobile sensors’ data satisfy

the user-defined predicate. Multiple mobile sensors may become centers of various target areas (called *query areas*), and they are defined as *reference sensors*. The set of reference sensors may keep changing because different mobile sensors may detect the event every sample period. As a result, query areas change according to the set of reference sensors as well as their locations. We categorize this type of queries as Event-based Location Dependent Query (ELDQ) which provides a fine granularity query model that only collects and aggregates the data from the sensor nodes in continuously changing query areas instead of the whole sensor network. ELDQ generalizes many typical query types in WSNs. For instance, a typical aggregation query is an ELDQ in which query areas are fixed, and a location dependent query is an ELDQ without event detection and aggregation.

In fact, there are several challenges in processing ELDQs: (1) Accessing mobile sensors in a robust and efficient manner in mobile WSNs. Although ZebraNet [18] has addressed the problem of tracking and accessing mobile sensors, its system model and design goals differ from ours. We need to design a scheme for efficiently disseminating queries to mobile sensors by stationary sensor infrastructure while minimizing the total cost. (2) Determination of continuously changing query areas. In determining up-to-date query areas, the centralized methods [1, 2, 3, 6] incur significant transmission overhead and latency, because query areas continuously change in a distributed manner with mobile sensors’ locations and the event detected each sampling period. (3) Efficient data aggregation in continuously changing query areas. Previous in-network aggregation methods [1, 2, 3] cannot efficiently aggregate data in continuously evolving areas. (4) Multiple concurrent queries may cause message collisions and system overload.

Motivated by the above challenges, we propose a set of techniques to process ELDQs in mobile WSNs. Stationary sensors are first built into a tree structure with the base station as root. User’s query is sent to the base station which then disseminates the query to each mobile sensor using the sensor tree. After receiving the query, mobile sensors keep monitoring the event every sample period. Mobile sensors that detect the event become reference sensors and start to propagate the query to the stationary sensors in the query areas. The stationary sensors that receive the query start to collect and aggregate the required

data. They adaptively choose routes to transmit aggregation results back to the base station according to current query areas. Such adaptation greatly reduces the transmission cost and query response time. Note that there are possibly multiple concurrently running queries issued to the base station. They are optimized both at the base station and at the mobile sensors to reduce the total processing cost. We make the following contributions in this paper:

- To our best knowledge, it is the first mechanism to define and process event-based location dependent queries in mobile WSNs. An ELDQ is a general query which exists in many applications and can not be efficiently answered by previous work.
- For efficient query dissemination to mobile sensors, we propose an adaptive proxy selection algorithm to minimize the update cost of the *proxy route* (defined in Section 4.2), when the mobile sensors change their proxies.
- In-network query propagation and Location-based In-network Aggregation (LIA) algorithms are proposed to efficiently process ELDQs.
- To optimize multiple queries, a *two-level multi-query optimization* algorithm that filters hidden queries at the base station and instantly rewrites concurrently triggered queries at each mobile sensor is proposed.
- We give a theoretical analysis and an extensive evaluation of the ELDQ processing performance. ELDQ exhibits a superior performance in terms of energy efficiency, query latency and query accuracy under various network conditions and outperforms all compared techniques.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 gives the system model and query definition. Section 4 presents a set of techniques for processing ELDQs. In section 5, we analyze the theoretical cost of processing ELDQ. Section 6 proposes a two-level multi-query optimization algorithm. The ELDQ processing performance is evaluated through simulation in Section 7. Section 8 concludes the paper.

## II. RELATED WORK

Processing location queries over moving objects have been studied extensively in both centralized and distributed database system [4, 14]. However, these solutions can not be applied directly to process data service queries in WSNs with constrained resources and distributed topology.

The work on query processing in WSNs can be classified into two types: the solutions that consider the sensor mobility or not. Typical data aggregation approaches collect and aggregate data from stationary sensors within a fixed area. A well-known example is ACquisitional Query Processing (ACQP) in sensor networks based on TinyDB [2]. Tiny Aggregation Service (TAS) [1] gives an in-network solution to answer aggregation queries over stationary sensors where data are aggregated in the network and sent back through the routing tree to the base station. However, TAS can only answer simple and declarative aggregation queries. A recent

work [3] proposes a Two-Phase Self-Join (TPSJ) scheme to efficiently process self-join query for event detection in sensor networks. However, the above approaches do not consider sensor mobility and can not efficiently process location dependent queries.

Some recent work starts to address sensor mobility for query processing in mobile WSNs such as CNFS [15], CountTorrent [16], ICEDB [17] and MobiQuery [19]. However, these approaches either lack the in-network optimization [17, 19] or introduce additional overhead in query processing through maintaining a partial map of the network [15] and informing all the network nodes of the aggregate query result [16]. Moreover, their system models and query types are different from ours. For example, ELDQ is more challenging than the simple data collection query in CNFS and ICEDB that collects data from a certain source node or nodes in a fixed area and the generic aggregation query in CountTorrent that aggregates sensor data in a fixed area, because ELDQ needs to aggregate sensor data in a continuously changing query area. MobiQuery fails to optimize in-network processing cost for mobile spatialtemporal queries.

Multi-query optimization in WSNs is studied in some recent work including MQO [12] where communication cost is optimized based on complexity analysis and TTMQO [11] where the base station groups incoming queries into synthetic queries based on the cost model to save the total cost of query processing. However, these methods only consider simple aggregation queries without sensor mobility and do not take the unique properties of ELDQs into account such as distributed event detection and continuously changing query areas.

## III. PRELIMINARY

### A. System Model

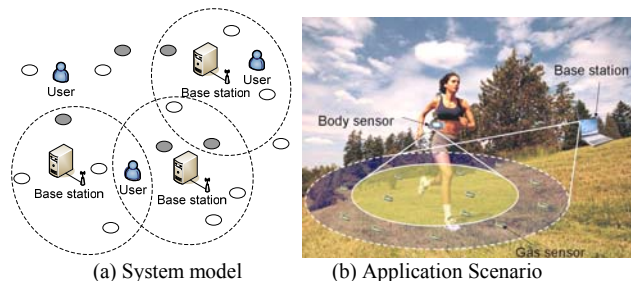


Figure 1. System model and scenario

We assume each stationary sensor has a unique identifier and is aware of its own location through positioning devices (e.g. GPS) or localization techniques like [8]. There is a low-level mechanism, such as beacon messages, that enables each stationary sensor to know its neighbors. Mobile sensors' current velocities and locations are measured by themselves using GPS. In Figure 1(a), the whole area is divided into several sub areas; each sub area has a base station that is responsible for injecting user's queries to the sensor network and sending the query results

back to the user. The white circles are the stationary sensors and the gray circles are the mobile sensors. In this paper, we leave the issues in inter-base station communication as future work and discuss the situation in one sub area. Figure 1(b) shows an application scenario of the system. The small circle is the mobile body sensor's transmission range, and the large circle is the query area. In this scenario, the body sensor keeps monitoring the running person's health status. When it detects the person's blood oxygen level is lower than a threshold, it triggers and propagates the query to the gas sensors in the query area. Those gas sensors aggregate the air pollutants data and report the results back to the base station.

### B. Query Definition

ELDQ is defined as the following format:

```
SELECT [aggregation operators] S1.att1... S1.atti
FROM Sensor1 AS S1
INSIDE (Radius, (SELECT S2.loc
FROM Sensor2 AS S2))
START ON EVENT1(S2.atti)
STOP ON EVENT2(S2.atti)
SAMPLE PERIOD Sp
FOR Duration
```

where aggregation operators include SUM, AVG, MAX, MIN and COUNT; att<sub>i</sub> is the data type of the sensor,  $1 \leq i < n$ , where n is the maximum number of data types a sensor can sample; EVENT<sub>1</sub> represents the triggering conditions of the query and EVENT<sub>2</sub> represents the stopping conditions of the query; sample period and duration means mobile sensors should detect the event each sample period for a specific duration. As presented in [4], most of the location constraints can be expressed in terms of INSIDE, so only INSIDE is considered in the query. We only discuss the processing of the ELDQs with aggregation operator AVG, because such ELDQs requires not only the data but also the number of all the sensors in the query area thus is usually the most challenging query type. In fact, our mechanism can also efficiently process the ELDQs with other aggregation operators.

## IV. PROCESSING ELDQS

In this section, we discuss processing an ELDQ in a distributed and concurrent way to reduce the total processing cost. Stationary sensors are first built into a tree structure with the base station as root similar to the approach in [6]. As a result, each stationary sensor's level in the tree is proportional to the transmission hops from the base station to itself, which minimizes the number of sensors to be traversed in query dissemination. Each stationary sensor maintains a Minimum Bounding Rectangle (MBR) which bounds the maximum extents of all its child nodes' location in the tree [9]. The overlapping areas among MBRs are minimized to decrease the number of messages for accessing sensor nodes. Each stationary sensor also maintains a neighbor stationary sensor nodes list including each neighbor's id and location. The above information is

updated by each stationary sensor using periodical beacon messages. Note that the topology of stationary sensors does not evolve frequently, so the cost of maintaining such information is comparatively low. The ELDQ processing is divided into four main steps: query dissemination, event detection, query propagation and data aggregation.

### A. Dissemination to Mobile Sensors

To reduce the cost, the base station disseminates the query to mobile sensors using the tree structure instead of flooding as in [5]. Each mobile sensor registers to a stationary sensor, called *proxy*, to access the sensor network. The base station assigns each incoming query a unique qid and converts it into the query item (qid, attribute list, aggregation operators, events, radius, sample period, duration) and disseminate the query item to each mobile sensor along a route from the base station to the mobile sensor's proxy (called *proxy route*). Each stationary sensor in the proxy route caches the next hop to the mobile sensor's proxy, so it only needs to unicast the query item to the next hop, which greatly reduces the number of dissemination messages. However, the mobile sensor changes its proxy frequently due to mobility. As a result, the proxy route should be updated by adding the mobile sensor's proxy information to the new proxy route and deleting the out-of-date proxy information from the obsolete proxy route. We argue that the number of update messages can be reduced if mobile sensors properly select their new proxies.

An extreme situation may happen during the period that the obsolete proxy route has not been deleted immediately after a mobile sensor change its proxy. A query may be disseminated along the obsolete proxy route to the old proxy which has already lost contact with the mobile sensor. In this case, after the old proxy receives the update message, it sends the query up along the old proxy route until to the node that caches the next hop to the new proxy. Then the query is disseminated along the new proxy route to the mobile sensor. To avoid message losses, each sensor in the proxy route overhears its next hop. If a parent node does not overhear the query transmitted by the next hop, it assumes the next hop has not received the query, and retransmits the query.

### B. Adaptive Proxy Selection Algorithm

Each mobile sensor checks the connectivity with its proxy by sending periodical handshake messages to its proxy. When a mobile sensor is out of contact with its proxy, it needs to select a new proxy from its neighbor stationary sensors.

*Definition 1:* Consider the base station's id is  $S_0$ , a stationary sensor  $S_i$ 's *dissemination path* is  $(S_0, \dots, S_i)$  which is the order of traversing sensors from the base station to  $S_i$  through the tree.

*Definition 2:* Consider stationary sensor nodes  $S_i$ 's dissemination path is  $(S_0, \dots, S_k, \dots, S_i)$  and  $S_j$ 's dissemination path is  $(S_0, \dots, S_k, \dots, S_j)$  where  $S_k$  is the overlapping node of the two dissemination paths, the *tree distance* between  $S_i$

and  $S_j$  is  $|S_k, \dots, S_i| + |S_k, \dots, S_j| - 2$ .  $|S_k, \dots, S_i|$  is the number of sensor nodes on the path.

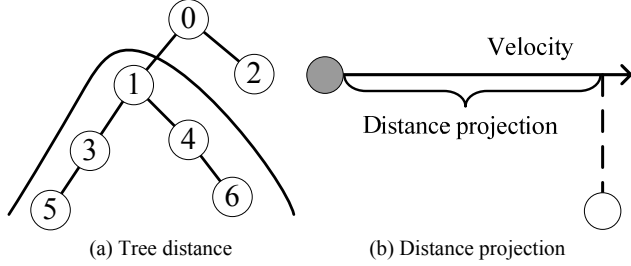


Figure 2. Various distances

In Figure 2(a), the dissemination paths for sensor node 5 and 6 are (0, 1, 3, 5) and (0, 1, 4, 6) respectively, so tree distance between them is 4. When a mobile sensor changes its proxy (e.g. from 5 to 6), the proxy route should be updated by relaying the mobile sensor's information from the new proxy to the overlapping node of the new proxy route and the old proxy route (it is node 1 in Figure 2(a)). Then the obsolete proxy route from the overlapping node to the old proxy is deleted. Therefore, the number of update messages during each proxy changing depends on the tree distance between the new proxy and the old proxy. We observe that the total update cost for a mobile sensor is affected by the number of proxy changing in the query's duration and the average tree distance between the mobile sensor's old proxies and new proxies, and it can be calculated as:

$$C_u = C_{tr} \sum_{i=1}^{N_{pc}} TD_i. \quad (1)$$

where  $C_{tr}$  is the transmission cost of each message,  $N_{pc}$  is the total number of proxy changing, and  $TD_i$  is the tree distance between the old proxy and the new proxy in the  $i$ th proxy changing. We propose two proxy selection algorithms to reduce  $C_u$  in two directions: Maximum Distance Projection (MDP) aims to reduce  $N_{pc}$  while Minimum Tree Distance (MTD) aims to minimize  $TD_i$ .

As shown in Figure 2(b), distance between a mobile sensor (the gray circle) and a stationary sensor is projected on the mobile sensor's velocity vector. This projection is called *distance projection*. In MDP algorithm, each mobile sensor registers to the neighbor stationary sensor that has maximum distance projection on its velocity direction. As a result, mobile sensor may stay in the proxy's transmission range for longer time, which reduces  $N_{pc}$  in the query's duration. When a mobile sensor starts to change its proxy, it broadcasts its current location and velocity to its neighbor stationary sensors. Each neighbor stationary sensor calculates its distance projection with the mobile sensor locally and sets a timer that is inversely proportional to the distance projection. The neighbor with maximum distance projection first sends its distance projection to the mobile sensor and is selected as the mobile sensor's new proxy. Then the mobile sensor replies a register message to the new proxy. Other neighbor stationary sensors overhearing

this message will not send their distance projections to the mobile sensor to save the transmission cost.

In MTD algorithm, each mobile sensor registers to the neighbor stationary sensor that has minimum tree distance to its old proxy. Each stationary sensor keeps a dissemination path that can be obtained in the sensor tree's building procedure by recording the ids of the sensors along the traversing path from the base station to itself. Therefore, obtaining dissemination path does not introduce extra cost. When a mobile sensor needs to change its proxy, it broadcasts its old proxy's dissemination path to the neighbor stationary sensors. Each neighbor stationary sensor calculates its tree distance from the old proxy and set a timer proportional to the tree distance. The neighbor node with the shortest tree distance first sends the tree distance to the mobile sensor and is selected as the mobile sensor's new proxy. However, multiple neighbor sensors may have the shortest tree distance from the old proxy at the same time. The tie is solved by selecting the proxy that has minimum distance with the mobile sensor (MTD&MD) or selecting the proxy that has maximum projection distance with the mobile sensor (MTD&MDP). We rely on MAC layer to avoid collisions between the neighbor sensors with the same tree distance.

We have compared MDP, MTD&MD and MTD&MDP with the typical minimum distance (MD) algorithm [7] by simulation in GloMoSim where MAC layer protocol is set to IEEE 802.11. In MD algorithm, each mobile sensor selects the nearest stationary sensor as its new proxy. In our simulation, 200 stationary sensor nodes and 3 mobile sensor nodes are uniformly distributed in a square area as the mobile sensors' mobility pattern using *random way point* with pause time 30 seconds. The simulation time is set to 40 minutes and each mobile sensor node checks the connection with its proxy every 10 seconds.

We design two experiments with different node densities and sensors' transmission ranges. In the first experiment, we set the simulation area to 600m×600m and each mobile or stationary sensor's transmission radius to 100m. From Figures 3(a) and 4(a), we can see that MDP results in the minimum number of proxy changing and update messages. The advantage becomes more obvious when mobile sensors' maximum speed increases. Because MDP algorithm prolongs each mobile sensor's resident period within its proxy's transmission range thus reduces the number of proxy changing in the query's duration. In the second experiment, we change the simulation area to 1000m×1000m and each sensor's transmission radius to 200m. However, we can see from Figures 4(a) and 4(b) that although MDP minimizes the number of proxy changing, MTD&MDP instead of MDP results in the minimum number of update messages. As the sensor's transmission radius increases from 100m to 200m, each mobile sensor stays within its proxy's transmission range for a longer time, which results in less saving of the number of proxy changing by MDP. We can see from Figure 3 that the maximum saving of proxy changing by MDP decreases from 49 (Figure 3(a)) to 17 (Figure 3(b)). As depicted in (1), the total update cost for a

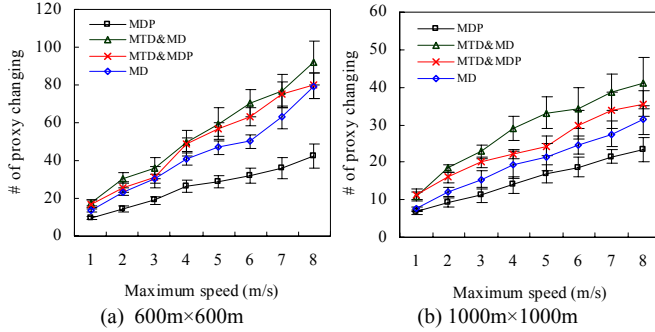


Figure 3. Num of proxy changing vs. Max speed

mobile sensor is determined by two factors: the number of proxy changing and the tree distance every proxy changing. If the saving of the number of proxy changing by MDP has decreased to certain extent, the saving of the number of update messages (equal to tree distance) every proxy changing becomes a dominant factor that determines the total update cost. That is the reason MTD&MDP becomes the optimal algorithm as each sensor's transmission radius increases to 200m. Note that MDP, MTD&MDP and MTD&MD always outperforms MD algorithm which leads to frequent updates and excessive update messages.

We learn from the above evaluations that different system scenarios may lead to different optimal proxy selection algorithms between MDP and MTD&MDP. We introduce an adaptive proxy selection algorithm to choose the optimal algorithm in certain system scenario. To decide the optimal algorithm, there is a training period after the sensor tree is built in which each mobile sensor runs both algorithms and records the respective total tree distance (equals to the total update messages). The algorithm that results in the minimum total update messages is chosen as the optimal one. Such adaptive scheme can optimize the total update cost for proxy changings. The query's duration is usually much longer than the duration in the simulation, so the algorithm's optimization effects are significant.

### C. Event Detection

Each mobile sensor performs event detection locally in a distributed and concurrent way. After receiving a query item, each mobile sensor keeps sensing the required data and checking the event every sample period for a specific duration. Once the event is detected, the mobile sensor becomes the reference sensor, and its current location becomes the center of the circle query area. Distributed event detection saves communication cost and enables earlier detection of events compared to the centralized approach where the base station is responsible for event detection [10].

### D. In-network Query Propagation

After a query is triggered by an event, the reference sensor propagates the *triggered query* including the reference sensor's id and current location, query id, query radius and attribute list to the stationary sensors in the query area. If the query radius is shorter than the reference

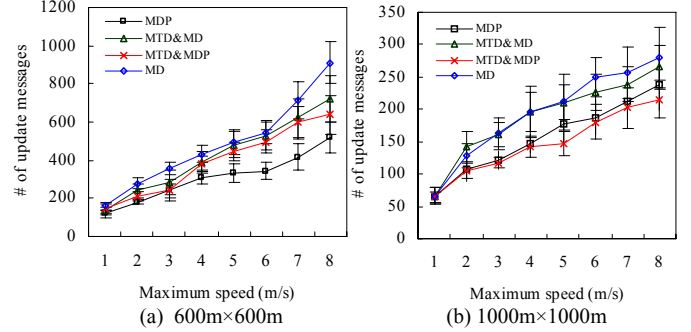


Figure 4. Num of update messages vs. Max speed

sensor's transmission radius, the triggered query only needs to be broadcasted to the stationary sensors in the query area by one message. Otherwise, the triggered query should be propagated to all the stationary sensors in the query area in a multi-hop fashion. In centralized methods, queries are propagated from the base station to the sensors in the query area by flooding [11] or tree-structured indexes [2, 3, 6], which wastes propagation messages and time. We propose an in-network query propagation approach based on each stationary sensor's local decision.

Each reference sensor first broadcasts the triggered query to its neighbor stationary sensors. Consider a stationary sensor node  $S$  receives the triggered query. If  $S$ 's parent node is not in the reference sensor's transmission range, it forwards the triggered query to its parent node, because its parent node can not receive the query directly from the reference sensor.  $S$ 's parent node can help to propagate the triggered query to  $S$ 's siblings that are isolated from other sensors in the query area. If  $S$  is not a leaf node, it broadcasts the triggered query to both its parent node and its child nodes that are out of reference sensor's transmission range and their MBRs overlap the query area. However, multiple child nodes may transmit the triggered query to the same parent node. If  $S$  overhears that one of its siblings has sent the triggered query to its parent nodes, it will not transmit the query to its parent node to avoid duplicate transmissions. The triggered query will be eventually sent back to the base station. If the base station does not receive the triggered query from one of its immediate child node  $S_i$  while it finds that  $S_i$ 's MBR overlaps the query area, the base station will inject the triggered query to  $S_i$ 's subtree. The reason is that some child nodes of  $S_i$  may be in the query area but would not receive the triggered query.

*Theorem 1:* Using in-network propagation, if a triggered query is received by at least one stationary sensor, it will be propagated to all the stationary sensors in the query area.

*Proof:* Assume sensor node  $S_i$  receives the triggered query. In our approach, triggered query is sent to sensor node's parent after being received. For any sensor node  $S_j$  in the query area, triggered query will be sent from  $S_i$  to the lowest common ancestor of  $S_i$  and  $S_j$  in the tree,  $S_k$ . Since  $S_k$ 's MBR overlaps  $S_j$ 's MBR,  $S_k$  will send the triggered query to  $S_j$ , which guarantees the correctness of Theorem 1.

### E. Location-based In-network Aggregation

In generic in-network aggregation algorithms [1, 2, 3], each sensor always transmits the data to its parent node. When query areas keep changing, many transmissions will be wasted because some parent nodes are not in the query areas thus can not perform local aggregation. Note that each sensor in the query area should transmit at least one message since it needs to send its data. However, sensors out of the query area usually do not need to send any messages except relaying the aggregated data to the base station. Thus, transmission cost can be saved by reducing the number of sensors that participate in the aggregation and are not in the query areas.

We propose a Location-based In-network Aggregation (LIA) algorithm to enable each sensor to dynamically choose the next hop sensor to transmit data based on the reference sensor's current location. A sensor node transmits data to its parent node in either of the two situations: (1) parent node is in the query area; (2) no neighbor nodes with equal or higher level are in the query area, except the neighbor nodes that transmit data to this sensor. Otherwise, the sensor node transmits the data to the neighbor node with the highest level in the query area. If there are more than one such neighbor nodes, the sensor node choose the nearest one to guarantee the link quality. As a result, fewer sensor nodes are involved in the aggregation and more data are aggregated locally.

Each non-leaf sensor node should wait until it has heard from its child nodes in the query area before aggregating and forwarding the aggregation results. It is possible that several sensor nodes at the same level transmit data to each other at the same time. To avoid the collisions, each non-leaf sensor's wait interval is set to be proportional to the number of its child nodes in the query area. Because sensor nodes with more child nodes in the query area may aggregate more sensors' data locally, they need a longer interval to wait for other sensors' data to save transmission cost. Since each sensor has a neighbor list containing each neighbor node's id and location, it can calculate the distance between each child node's location and the reference sensor's current location which has been received in the query propagation. If the distance is shorter than the query radius, the current child node is in the query area. After scanning the whole neighbor list, a sensor can locally get the up-to-date number of its child nodes that are in the query area. Assume the maximum depth of the sensor tree is  $d$ , the non-leaf sensor node is at level  $l$  and has  $i$  child nodes in the query area ( $0 \leq i \leq n$ ), where  $n$  is the maximum number of child nodes in the tree. If each sensor's sensing and processing time is  $T_{sp}$  and transmission time is  $T_t$ , the sensor's wait interval  $WI$  can be calculated as:

$$WI = (d - l)(2 \times T_t + T_{sp}) + i \times T_t \quad (2)$$

$WI$  considers the longest hops  $d - l$  that the non-leaf sensor node forwards the triggered query to the leaf nodes in the query area which then transmit their data back to it.

After receiving the triggered query, each non-leaf sensor in the query area calculates its own wait interval  $WI$  and transmits the aggregation result with the number of sensors

that contribute to the result to the next hop after  $WI$  expires, while each leaf sensor in the query area transmits its data to the next hop without waiting. The sensors out of the query area send the aggregation results to the next hops immediately after receiving them. These sensor nodes continue to aggregate and forward the data in this manner, until the query results arrive in the base station.

### V. COST ANALYSIS

Since energy consumption of the sensor network is dominated by radio transmissions, we give a theoretical analysis of the transmission cost of ELDQ processing ( $cost(Q)$ ) including the cost of dissemination ( $diss(Q)$ ), propagation ( $prop(Q)$ ) and aggregation ( $agg(Q)$ ).

$$cost(Q) = diss(Q) + prop(Q) + agg(Q) \quad (3)$$

An ELDQ  $Q$  can be decomposed into attribute set  $Q.att$ , event  $Q.e$ , sample period  $Q.sp$ , radius  $Q.r$  and duration  $Q.d$ . Assume  $N$  stationary sensor nodes and  $N_m$  mobile sensor nodes uniformly distributed in a circle area with radius  $R_{max}$  and the base station at its center.

$Q$ 's dissemination cost includes the cost of query dissemination and proxy changing. Assume the average query arrival time is  $t$  seconds per query,  $diss(Q)$  is:

$$diss(Q) = C_{tr} \left( \frac{Q.d}{t} \sum_{j=1}^{N_m} (l(M_j) + 1) + \sum_{i=1}^{N_{pc}} TD_i \right) \quad (4)$$

where  $l(M_j)$  is the level of mobile sensor  $M_j$ 's proxy.

The probability density of any location within the circle area is  $1/\pi R_{max}^2$ . Therefore, the distance  $R$  between the mobile sensor (the center of the circle in Figure 6) and the base station (point  $b$  in Figure 6) has the following probability distribution:

$$P(R \leq r) = \int_0^{2\pi} \int_0^r \frac{1}{\pi R_{max}^2} r' dr' d\theta = \frac{r^2}{R_{max}^2}$$

Thus the probability density function of  $R$  is:

$$f_R(r) = \frac{\partial P}{\partial r} = \frac{2r}{R_{max}^2}, 0 \leq r \leq R_{max}$$

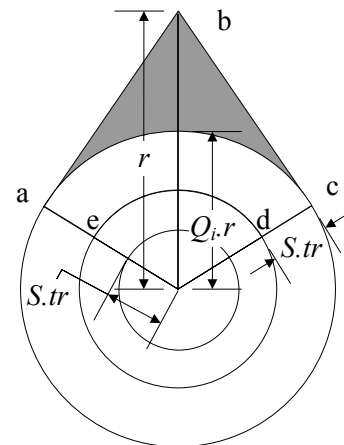


Figure 5. Theoretical model

If  $Q.r$  is shorter than the mobile sensor's transmission radius,  $prop(Q)$  is 0. Otherwise,  $prop(Q)$  includes the cost of propagation within the query area and propagation from *boundary nodes* to the base station. Boundary nodes are the sensor nodes in the query area whose parents are not in the query area. The number of propagation messages within the query area can be approximated to the number of sensors in the ring area between the mobile sensor's transmission range and the query area. As shown in Figure 5, we can regard the number of propagation messages from boundary nodes to the base station as the number of sensor nodes in the shadowed area *abc*. The number of propagation messages out of this area is relatively small. The expectation of the area of *abc* is:

$$\begin{aligned} E(Area_{abc}) &= \int_{Q.r}^{R_{\max}} \left( Q.r \sqrt{r^2 - Q.r^2} - Q.r^2 \arccos \frac{Q.r}{r} \right) \frac{2r}{R_{\max}^2} dr \\ &= \frac{Q.r}{3} \sqrt{R_{\max}^2 - Q.r^2} \left( \frac{Q.r^2}{R_{\max}^2} + 2 \right) - Q.r^2 \arccos \frac{Q.r}{R_{\max}} \\ &\leq Q.r \sqrt{R_{\max}^2 - Q.r^2} - Q.r^2 \arccos \frac{Q.r}{R_{\max}} \end{aligned}$$

Given the node density  $N/\pi R_{\max}^2$  and the sensor's transmission radius  $S.tr$ , the propagation cost of  $Q$  is then:

$$\begin{aligned} prop(Q) &= C_{tr} p(Q.e) (Q.d/Q.sp) (N/\pi R_{\max}^2) \\ &\quad (E(Area_{abc}) + \pi(Q.r^2 - S.tr^2)), \quad Q.r > S.tr \quad (5) \end{aligned}$$

where  $p(Q.e)$  is the probability that the event happens during a sample period.

$Q$ 's aggregation cost includes the cost of aggregation within the query area and transmitting aggregation results out of the query area. Aggregation results are aggregated in several boundary nodes at the highest level in the query area before they are transmitted back to the base station. Thus, most of such boundary nodes are in the ring area of *acde* in Figure 6 with  $S.tr$  as its width. The expectation of the area of *acde* is:

$$\begin{aligned} E(Area_{acde}) &= \int_{Q.r}^{R_{\max}} \frac{2r}{R_{\max}^2} (Q.r^2 - (Q.r - S.tr)^2) \arccos \frac{Q.r}{r} dr \\ &= (2Q.rS.tr - S.tr^2) \left( \arccos \frac{Q.r}{R_{\max}} - \frac{Q.r}{R_{\max}} \sqrt{1 - \frac{Q.r^2}{R_{\max}^2}} \right) \\ &< \frac{\pi}{2} (2Q.rS.tr - S.tr^2), \quad Q.r > S.tr \\ E(Area_{acde}) &= \int_{Q.r}^{R_{\max}} \frac{2r}{R_{\max}^2} Q.r^2 \arccos \frac{Q.r}{r} dr \\ &= Q.r^2 \left( \arccos \frac{Q.r}{R_{\max}} - \frac{Q.r}{R_{\max}} \sqrt{1 - \frac{Q.r^2}{R_{\max}^2}} \right) \\ &< \frac{\pi}{2} Q.r^2, \quad Q.r \leq S.tr \end{aligned}$$

The transmission ranges of all the boundary nodes that have aggregation results can cover the ring area *acde* with

little overlap. Thus, the number of these boundary nodes can be calculated as:

$$N_b = \left\lceil \frac{E(Area_{acde})}{\pi S.tr^2} \right\rceil$$

The expectation of the average level of these boundary nodes can be simply calculated as:

$$\begin{aligned} E(l_{avg}) &= \left\lceil \int_0^{R_{\max}} \left( \frac{r - Q.r}{S.tr} \right) \frac{2r}{R_{\max}^2} dr \right\rceil + 1 \\ &= \left\lceil \frac{2R_{\max} - 3Q.r}{3S.tr} \right\rceil + 1 \end{aligned}$$

In the LIA algorithm, each sensor node in the query area sends its data by one message. The number of sensor nodes in the query area is  $NQ.r^2/R_{\max}^2$ . Thus,  $agg(Q)$  is:

$$agg(Q) = C_{tr} p(Q.e) (Q.d/Q.sp) (NQ.r^2/R_{\max}^2 + N_b E(l_{avg})) \quad (6)$$

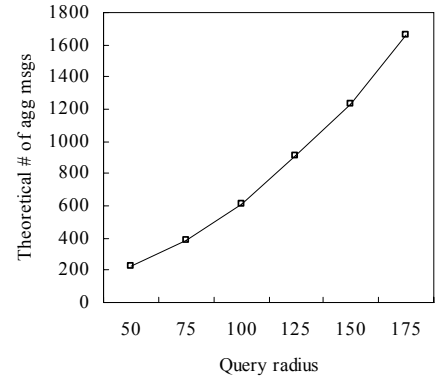


Figure 6. Theoretical num of agg msgs

The cost analysis suggests that our techniques minimize the redundant messages in processing ELDQs. From (4), we learn that dissemination cost is mainly affected by the mobile sensors' mobility pattern and the system scenario. The theoretical upper bounds of the cost of propagation and aggregation can be obtained from (5) and (6) and used to estimate the energy consumption of the WSNs. Even though our analysis is for an idealized case based on some assumptions, the simulation results in Section 7 match well with the theoretical cost. For example, the theoretical number of aggregation messages in Figure 6 is quite close to that in the simulation, most of the deviations are within 5% to 15% of the experimental value.

## VI. MULTI-QUERY OPTIMIZATION

In many applications, multiple concurrent queries may be issued to the base station. Processing multiple queries in an uncooperative manner will lead to bandwidth contention and transmission collisions. We propose a two-level multi-query optimization algorithm that filters *hidden queries* at the base station and instantly rewrites multiple concurrently triggered queries at each mobile sensor to reduce total processing cost.

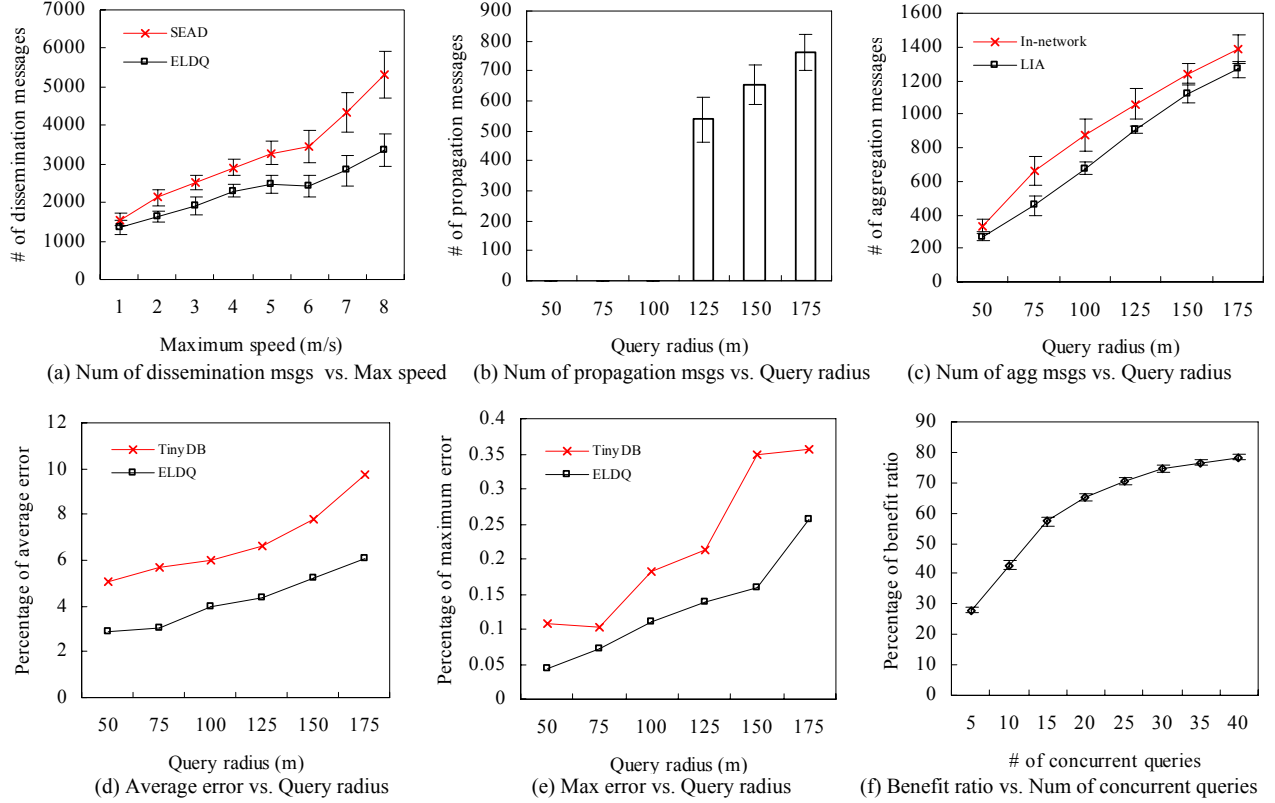


Figure 7. ELDQ Processing

### A. Optimization at the Base Station

A base station maintains a list of running queries. When an ELDQ  $Q_i$  arrives at a base station, our algorithm first scans the query list and compares each part of incoming query  $Q_i$  with the corresponding part of each running query  $Q$  in the query list. Assume a stationary sensor  $S$ 's attribute set is  $S.att$ , remaining duration of a running query  $Q$  is  $Q.rd$ .  $Q_i$  is a hidden query and need not to be injected to the sensor network if it satisfies all the following hidden rules: (1)  $Q_i.att \subseteq Q.att \subseteq S.att$ , (2)  $Q_i.e = Q.e$ , (3)  $Q_i.r = Q.r$ , (4)  $Q_i.sp \mid Q.sp$ , (5)  $Q_i.d \leq Q.rd$ . Rule (1) means  $Q_i$ 's attribute set is included in  $Q$ 's attribute set. Rule (2) guarantees that  $Q_i$ 's reference sensor set is the same as  $Q$ 's reference sensor set. By adding rule (3),  $Q_i$ 's query areas are the same as  $Q$ 's query areas. Rule (4) means  $Q_i$ 's sample period is divided by  $Q$ 's sample period. These hidden rules guarantee  $Q_i$ 's result set is totally included in  $Q$ 's result set and can be filtered out by the base station. If no query satisfies all the hidden rules with  $Q_i$ ,  $Q_i$  is added to the list of running queries and disseminated to mobile sensors.

We should consider the following two special cases. One special case is that  $Q_i$  satisfies rules (1) through (4) with  $Q$ ,  $Q_i$  will be rewritten to  $Q'_i$  with new a duration ( $Q_i.d - Q.rd$ ).  $Q'_i$  will not be injected to the sensor network until  $Q$  finishes, because it is possible that when  $Q'_i$  is waiting, another incoming query includes  $Q'_i$ 's results and

makes  $Q'_i$  unnecessary to be injected. Another special case is that  $Q_i$  satisfies rules (2) through (5) with multiple running queries in the query list. Assume  $Q_i$  satisfies rules (2) through (5) and the following rule with  $k$  ( $k \geq 2$ ) running queries  $Q_1, Q_2, \dots, Q_k$ : (1)\*  $Q_i.att \subseteq (Q_1.att \cup Q_2.att \cup \dots \cup Q_k.att) \subseteq S.att$ ,  $Q_i$ 's results can be obtained by drawing out the required attributes from the query results of  $Q_1, Q_2, \dots, Q_k$ . Rule (1)\* means  $Q_i$ 's attribute set is included in the union set of the attribute sets of  $Q_1, Q_2, \dots, Q_k$ , so  $Q_i$  also does not need to be injected to the sensor network. Optimization at the base station based on hidden rules can minimize duplicate access to the sensor network.

### B. On-line Query Rewriting Algorithm

Hidden rules are strict for many incoming queries and optimization at the base station does not exploit the similarities among queries at a fine granularity. We apply an *on-line query rewriting algorithm* at each mobile sensor. Compared to rewriting queries at the base station [11, 12], our algorithm can easily decide which queries are triggered every sample period, thus can intelligently rewrite the queries.

Each mobile sensor also keeps a list of running queries. After a mobile sensor receives an incoming query  $Q_i$ , it sets its own sample period at the greatest common divisor of the sample periods of all the queries.  $Q_i$ 's start time is set to be divisible by the mobile sensor's sample period. In this way,

queries will sample at the same time, and hence share the event detection. Although introducing such little delay will make the first sampling period start later, for a continuous query, this extra latency is acceptable. In each sample period of the mobile sensor, if multiple queries are triggered at the same time, the mobile sensor instantly rewrites these queries into a synthetic triggered query  $Q_{syn}$ .  $Q_{syn}$ 's attribute list is the union set of all the triggered queries' attribute list, and  $Q_{syn}$ 's query radius is set to the longest query radius of all the triggered queries. To maintain the semantic correctness,  $Q_{syn}$  should contain reference sensor's id and each triggered query's id which is mapped to the corresponding query radius. Then, the mobile sensor propagates  $Q_{syn}$  to the sensors in the query area instead of propagating the triggered queries separately. Each sensor that receives  $Q_{syn}$  sends its data and the id list of the triggered queries whose query areas it is in to the next hop sensor by LIA algorithm. The id list helps the next hop sensor in aggregating multiple results for multiple triggered queries. Rewriting and processing the multiple triggered queries in this way reduces the total cost because redundant transmissions are minimized by processing  $Q_{syn}$  while only introducing a little computation cost in the mobile sensor.

## VII. PERFORMANCE EVALUATION

We evaluate the performance of processing ELDQ through simulation using GloMoSim. As explained in the introduction, there are no other solutions that provide event-based location dependent data services, so we separately evaluate the techniques including query dissemination, in-network query propagation, location-based in-network aggregation and two-level multi-query optimization. We use the same setting as in Section 4.2 where the total area is 600m×600m and each sensor's transmission radius is 100m. The workload query has a sample period of 120 seconds and duration of 12000 seconds, and the probability that the event happens in each sample period is set to 10%, so the query is triggered around 10 times at each mobile sensor. Each data point in the figures has a 90% confidence interval which comes from the average result of 10 runs.

Dissemination in mobile WSNs has been studied in [5], [7] and [13]. However, IDDA [13] focuses on interest dissemination from the mobile sink to neighbor stationary sensors, which is opposite to the dissemination direction in our system. Moreover, the flooding approach in [5] is not efficient compared to the SEAD approach [7]. Thus, we compare our in-network query dissemination approach with SEAD. In SEAD, the mobile sink selects its nearest neighbor as its access node (MD algorithm) and sends a join query to the source which disseminates the data to the access node. We evaluate the total number of dissemination messages including the number of query dissemination messages and update messages caused by proxy changing. The average query arrival frequency is set to 5 minutes per query, thus there are around 40 queries being disseminated to the sensor network in 12000 seconds. As shown in Figure 7(a), our approach reduces the total dissemination

cost. The number of query dissemination messages is proportional to the average level of proxy sensor (correspond to access node in [7]), making the number of query dissemination messages of these two approaches quite close to each other. However, as discussed in Section 4.2, the MD algorithm used in SEAD results in more update messages compared to the adaptive proxy selection algorithm used in our approach. Note that total dissemination cost are largely affected by the number of update messages caused by proxy changing when the query arrive rate is not very high, so it is determined by the proxy selection algorithm in most of the cases. As the maximum speed of mobile sensors increases, the saving becomes more obvious.

We evaluate the total number of propagation messages in the query's duration. If the query radius is shorter than the mobile sensor's transmission radius, the number of propagation messages is 0 because sensors in the query area can receive the query from the mobile sensor directly. Otherwise, we can see from Figure 7(b) that the number of propagation messages increases with the query radius.

We compare the LIA algorithm with the generic in-network aggregation algorithm [1, 2, 3] in which each sensor aggregates the data of its child nodes in the tree structure. We set the maximum speed of mobile sensors to 1m/s. Figure 7(c) shows that LIA algorithm always outperforms the generic in-network aggregation algorithm in the number of aggregation messages. According to the results, the saving of messages increases and then decreases as query radius increases. The most saving happens around the query radius 75. As query radius increases, more sensors are involved in the aggregation, thus the saving of messages also increases. However, when the query radius is large enough that most of the sensors' parent nodes are in the query area, these sensors send their data to their parent nodes instead of dynamically choosing the destination. In this situation, only a part of the sensors in the query area save messages by using LIA algorithm. Message saving degrades if query radius increases above certain threshold. For a continuous query, the total saving will be significant when the query's duration is long enough in large scale sensor networks. It is interesting to note that the dissemination cost forms the main part of the ELDQ processing cost, which reveals that the mobility is a dominant factor that affects the total cost.

After evaluating each technique separately, we compare the accuracy of query results with that in TinyDB [2]. The accuracy is computed by dividing the error (i.e. deviation value) by the true value of the query results. Since TinyDB [2] cannot answer ELDQ properly, we apply in-network query dissemination and propagation approaches to TinyDB. The only difference is in aggregation part: generic in-network aggregation algorithm is used in TinyDB while LIA algorithm is used in our scheme. We evaluate the accuracy by measuring the percentage of the average error and maximum error. Figure 7(d) shows that query results are more accurate in our scheme, because fewer sensors are involved in the aggregation than TinyDB, reducing the

possibility of message losses. As the query radius increases, both the average error and maximum error increase because message losses increase if more sensors are involved in the query propagation and aggregation. In fact, the accuracy of the query results is affected by the messages losses in the query dissemination, query propagation and data aggregation. As discussed in Section 4, several techniques are employed to avoid message losses. However, in the hierarchical tree structure, a single node failure can result in the aggregated data of a sensor node and its neighbor nodes being lost, making the maximum errors highly variable in Figure 7(e).

We evaluate the benefit ratio of the two-level multi-query optimization algorithm with the number of concurrently running queries. The benefit ratio is computed by dividing sum of message savings by the sum of every query's messages. The workloads are set to 100 queries with average arrival frequency 60 seconds per query. These queries randomly choose their attribute lists (lights, temp), event probability (from 10% to 90%), query radii (from 25m to 175m), sample periods (from 50s to 500s). We vary the average duration to control the average number of concurrent queries. All the queries' sample periods and durations are divided by a smallest time unit 1 second, and their aggregation operators are set to AVG. As shown in Figure 7(f), the benefit ratio increases significantly from around 28% to 78% as the number of concurrently running queries increase from 5 to 40. This is because greater sharing can be exploited among more running queries.

### VIII. CONCLUSIONS

In this paper, we define a general query type Event-based Location Dependent Query (ELDQ) which exists in a wide range of data centric applications in mobile WSNs. Existing approaches can not efficiently answer ELDQs. We proposed a set of techniques to efficiently process ELDQs. We also give a theoretical analysis of the ELDQ processing cost. The cost analysis and experimental results show that these techniques can reduce the processing cost compared to the state-of-the-art approaches. These techniques can also be efficiently applied to typical query types in sensor networks.

In the future, we plan to extend our scheme in two directions. First, we want to design a new event detection mechanism to accurately detect sophisticated events. Second, we will adapt our scheme to optimize continuous processing cost in inter-base station scenario. These extensions will make our scheme even more efficient for answering ELDQs.

### ACKNOWLEDGEMENT

This research work was supported by NSF CNS-

0614886, KOSEF WCU Project R33-2008-000-10110-0 and National Pre-research Project of China 513150402.

### REFERENCES

- [1] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," in OSDI, 2002.
- [2] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: An Acquisitional Query Processing System for Sensor Networks," ACM Transactions on Database Systems, 2005.
- [3] X. Yang, H. B. Lim, M. T. Ozsu, and K. L. Tan, "In-Network Execution of Monitoring Queries in Sensor Networks," in SIGMOD, 2007.
- [4] S. Ilari, E. Mena, and A. Illarramendi, "Location-Dependent Queries in Mobile Contexts: Distributed Processing Using Mobile Agents," IEEE Transactions on Mobile Computing, vol. 5, 2006.
- [5] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A Two-Tier Data Dissemination Model for Largescale Wireless Sensor Networks," in Mobicom, 2002.
- [6] A. Soheili, V. Kalogeraki, and D. Gunopulos, "Spatial Queries in Sensor Networks," in GIS, 2005.
- [7] H. S. Kim, T. F. Abdelzaher, and W. H. Kwon, "Minimum-Energy Asynchronous Dissemination to Mobile Sinks in Wireless Sensor Networks," in Sensys, 2003.
- [8] R. Stoleru, J. A. Stankovic, and S. Son, "Robust Node Localization for Wireless Sensor Networks," in EmNets, 2007.
- [9] A. Guttman, "R-trees: A Dynamic Index Structure for Spatial Searching," in SIGMOD, 1984.
- [10] D. J. Abadi, S. Madden, and W. Lindner, "REED: Robust, Efficient Filtering and Event Detection in Sensor Networks," in 31st VLDB Conference, 2005.
- [11] S. Xiang, H. B. Lim, K.-L. Tan, and Y. Zhou, "Two-Tier Multiple Query Optimization for Sensor Networks," in ICDCS, 2007.
- [12] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman, "Multi-query Optimization for Sensor Networks," in DCOSS, 2005.
- [13] Y. Wu, L. Zhang, Y. Wu, and Z. Niu, "Interest Dissemination with Directional Antennas for Wireless Sensor Networks with Mobile Sinks," in Sensys, 2006.
- [14] B. Gedik, K.-L. Wu, P. S. Yu, and L. Liu, "Processing Moving Queries over Moving Objects Using Motion-Adaptive Indexes," IEEE Transactions on Knowledge and Data Engineering, 2006.
- [15] H. Huang, J. H. Hartman, and T. N. Hurst, "Efficient and Robust Query Processing for Mobile Wireless Sensor Networks," in GLOBECOM, 2006.
- [16] A. Kamra, V. Misra, and D. Rubenstein, "CountTorrent: Ubiquitous Access to Query Aggregates in Dynamic and Mobile Sensor Networks," in Sensys, 2007.
- [17] Y. Zhang, B. Hull, H. Balakrishnan, and S. Madden, "ICEDB: Intermittently-Connected Continuous Query Processing," in ICDE, 2007.
- [18] Juang, P., Oki, H., Wang, Y., Martonosi, M., et al, "Energy Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet," in ASPLOS, 2002.
- [19] C. Lu, G. Xing, O. Chipara, C.-L. Fok, and S. Bhattacharya, "A Spatiotemporal Query Service for Mobile Users in Sensor Networks," in the 25th IEEE International Conference on Distributed Computing Systems, 2005.