

# Event-triggered Location Aware Data Services in Mobile WSNs

Liang Hong<sup>1</sup>

Yafeng Wu<sup>2</sup>

Sang H. Son<sup>2</sup>

<sup>1</sup>College of Computer Science and Technology  
Huazhong University of Science and Technology  
Wuhan, Hubei, China 430074  
lh3zt@cs.virginia.edu

<sup>2</sup>Department of Computer Science  
University of Virginia  
Charlottesville, VA 22903  
{yw5s, son}@cs.virginia.edu

## ABSTRACT

Mobile *Wireless Sensor Networks* (WSNs) are increasingly deployed in many applications to provide various data services to users. However, existing data service approaches are incapable of efficiently answering some complicated queries in mobile WSNs. In this paper, we propose a new event-triggered query model which generalizes many location aware queries that continuously aggregate sensor data around mobile sensors of interests. We investigate difficulties to process such queries in mobile WSNs, and propose a complete set of techniques to answer such queries while optimizing system performance. These techniques include a novel tree structure for efficient access to mobile sensors and query propagation, a dynamic in-network aggregation scheme to reduce query answering cost, and an online query insertion algorithm to utilize sharing among multiple queries. The reasoning and simulation results demonstrate the promise of our strategy.

## Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems C.3 [Special-Purpose and Application Based Systems]: Real-time and embedded systems.

## General Terms

Algorithms, Performance, Design, Experimentation

**Keywords:** Wireless Sensor Networks, Data Service, Mobile Sensor Networks

## 1. INTRODUCTION

Wireless sensor networks are widely used in many applications to provide various data services [1, 8]. Users can issue declarative queries over sensor networks ranging from

simple data collection and aggregation [6, 9, 10] to event detection queries [1]. However, in mobile WSNs, some queries are complicated and cannot be efficiently answered by using existing approaches. In this paper, we consider a class of data service queries that collect the information around the mobile sensor nodes with timing constraints. Such queries require continuous monitoring as well as dynamic determination of target areas on the fly, and hence need specialized optimization techniques to be processed efficiently. An example query is “Give me the average percentage of air pollutants within 5 mile around the persons whose blood oxygen levels are lower than 90% sampled every 10 minutes in the next 10 days”. This type of query is triggered by the event that mobile sensors’ sampling data satisfy the predicate every sample period. Multiple mobile sensors may become location constraint’s reference objects which are defined as *reference sensors*. The set of reference sensors may keep changing because different mobile sensors may detect the event and become reference sensors every sample period. As a result, target areas (called *query areas*) change according to the set of reference sensors as well as reference sensors’ locations. We categorize this type of queries as *event-triggered location aware query (ETLAQ)*. ETLAQs represent a more general type of data service queries in WSNs compared to other typical query types. For example, aggregation query is an ETLAQ that query areas are fixed, and location aware query is an ETLAQ without event detection and aggregation. There are several challenges in processing ETLAQs that take sensors’ mobility into account:

1. Accessing mobile sensors. Although ZebraNet [8] has addressed the problem of tracking mobile sensors in a specific area, we need to design a scheme suitable for disseminating queries to mobile sensors in a WSN consisting of both stationary and mobile sensors.

2. Determination of query area. Since query areas continuously change with mobile sensors’ data and location, one of the key challenges is determining query areas and requiring all the sensors in those areas to sample and transmit data every sample period.

3. Dynamic in-network aggregation. Prior works [1, 6] on in-network aggregation and typical centralized

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*HotEmNets’08*, June 2–3, 2008, Charlottesville, Virginia, USA.  
Copyright 2008 ACM 978-1-60558-209-2/08/0006 ...\$5.00.

aggregation methods [9, 10] cannot efficiently support aggregation in continuously changing query areas. It is challenging to minimize the transmission cost of in-network aggregation in such query areas.

Motivated by these challenges, we propose an efficient and distributed event detection and in-network aggregation strategy to provide event-triggered location aware data services in mobile WSNs. We assume that mobile sensors and some stationary sensors have GPS to locate themselves; other sensors' location can be obtained through localization. The whole area is divided into several areas; each area has a base station that handles the queries associated with sensors in the area. Stationary sensors are first built into a novel tree structure with the base station as root to minimize the cost of query dissemination and facilitate in-network aggregation. The base station disseminates a query to each mobile sensor through the tree. After receiving the query, mobile sensors keep monitoring the event every sample period. Mobile sensors that detect the event become reference sensors. Then they start to determine query areas and require stationary sensors in the query areas to collect and aggregate data. Event detection and determination of query areas are performed in a distributed and concurrent way to save transmission cost and reduce query response time. Sensors adaptively choose routes to transmit aggregation results back to the base station according to the query areas. Such adaptation greatly reduces the transmission cost. 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. Thus, it is necessary to perform multi-query optimization to increase the scalability of our strategy. The contributions of this paper are:

1. To the best of our knowledge, it is the first approach to provide event-triggered location aware data services in mobile WSNs.
2. We design a novel tree structure for efficient query dissemination to mobile sensors and in-network aggregation. A *minimal tree distance* update method is proposed to reduce update messages when mobile sensors change their locations.
3. We propose a dynamic in-network aggregation algorithm for each stationary sensor that adaptively chooses the route to transmit data back to the base station according to continuously changing query areas.
4. We propose an online query insertion algorithm including hidden rules and compatible rules. This algorithm reduces the total cost for processing multiple ETLAQs.

The rest of this paper is organized as follows. Section 2 presents an algorithm on forming QA-tree. In section 3, we propose an in-network query processing strategy with multi-query optimization. Related works are discussed in section 4. Section 5 concludes the paper.

## 2. QA-TREE

We propose *query dissemination and aggregation tree (QA-tree)* for efficient dissemination of queries from a base station to each mobile sensor and in-network aggregation. QA-tree consists of both stationary sensors and mobile sensors. Considering the challenges given in Section 1, we set three optimization goals for QA-tree:

1. Minimal hops of query dissemination. To minimize query dissemination hops, each sensor's level in QA-tree is proportional to the hops of transmission from the base station to the sensor.
2. Minimal overlaps of MBRs. Each stationary sensor keeps a *minimal bounding rectangle (MBR)* of its child nodes to facilitate query dissemination. A node's MBR bounds the maximal extents of all its child nodes' location in the tree. The overlaps of MBRs cause the query being disseminated to more sensors that are not involved in the query.
3. Minimal update messages. In QA-tree, each mobile sensor registers to a stationary sensor, called *proxy*, to access the WSN. It will be costly if a base station disseminates the query to mobile sensors by flooding. QA-tree maintains routes from base station to mobile sensors (called *mobile sensor routes*) to reduce the query dissemination cost. Each stationary sensor in a mobile sensor route caches the next hop to the mobile sensor. Queries are relayed by each stationary sensor along the route to the mobile sensor. However, mobile sensors change their proxies frequently due to mobility. As a result, related mobile sensor routes should be updated. We argue that update messages can be reduced if mobile sensors properly choose their new proxies.

The first two optimization goals can be achieved by forming stationary sensors into a tree structure with the base station as root similar to that in [7]. The key difference is that each sensor node has not only a level in the tree and child nodes' MBRs but also a list of neighbor nodes' location and a *route path* from base station to itself. Based on neighbor nodes lists, stationary sensors adaptively choose the optimal route to transmit their aggregation results back to the base station. A route path is used in proxy choosing.

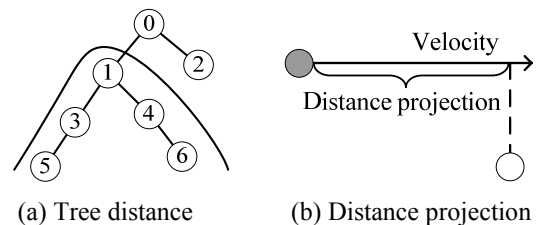
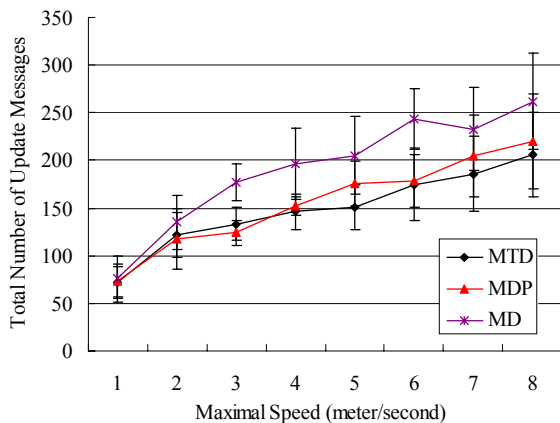


Figure 1. Various distances

Each mobile sensor sends handshake messages to its proxy periodically to make sure that it does not move out of proxy's radio range. When a mobile sensor is out of contact with its proxy, it uses the *minimal tree distance (MTD)*

method to choose a new proxy. Tree distance is the number of hops between two sensor nodes in QA-tree which can be calculated using route paths. In Figure 1(a), the route paths for 5 and 6 are 0-1-3-5 and 0-1-4-6 respectively (assuming node 0 is base station), so tree distance between the two nodes is 4. The mobile sensor first broadcasts the old proxy's route path to neighbor nodes. Each neighbor node calculates the tree distance from the old proxy, then sends it to the mobile sensor which will register to the neighbor stationary sensor with minimal tree distance as new proxy. After the mobile sensor changes its proxy, the mobile sensor route should be updated by relaying mobile sensor's information from new proxy to the overlap node of new proxy's route path and old proxy's route path (it is node 1 in Figure 1(a)). Intermediate sensors on the relay path cache the route to the mobile sensor. Then the obsolete route to the mobile sensor in each stationary sensor from the overlap node to old proxy is deleted. Note that the number of update messages depends on the tree distance between the new proxy and the old proxy. Therefore, the MTD method can minimize update messages when mobile sensors change their proxies.

We have compared MTD with other methods for choosing a new proxy by simulation: *minimal distance (MD)*, *highest level (HL)*, and *maximal distance projection (MDP)*. In MD, each mobile sensor registers to its nearest stationary sensor, solving the tie by choosing the sensor with the highest level. In HL, each mobile sensor registers to neighbor sensor with the highest level, solving the tie by choosing the nearest sensor. In MDP, each mobile sensor registers to the stationary sensor that has maximal distance projection between them to reduce the number of proxy changing. In Figure 1(b), distance between a mobile sensor (the gray circle) and a stationary sensor is projected on the mobile sensor's speed vector. This projection is called *distance projection*.



**Figure 2. Update cost with varying maximal speed.**

In our simulation, 200 stationary sensor nodes and 3 mobile sensor nodes are uniformly distributed in a  $1000m \times 1000m$  area, each node's radio range is  $200m$  and its mobility pattern is *random way point* with pause time 30 seconds. The

simulation time is 40 minutes and each mobile sensor node checks the connection to its proxy every 10 seconds. From Figure 2, we can see that MTD method uses the least number of update messages compared to MDP and MD methods. The advantage becomes more obvious when mobile sensors' maximal speeds increase. The reason is that although MDP method leads to less number of proxy changes than MTD, it usually causes more update messages than MTD in one proxy change and MD method usually leads to frequent updates. The query's duration is usually much longer than the duration in the simulation, so MTD's optimization effects are considerable. HL method is not shown in Figure 2 because it causes much more update messages than other three methods.

### 3. QUERY PROCESSING

#### 3.1 In-network ETLAQ Processing

In this section, we discuss answering ETLAQ using QA-tree in a distributed and concurrent way. We mainly focus on reducing the transmission cost and the query response time.

##### 3.1.1 Query dissemination

Previous query dissemination approaches [1, 6, 9] hardly consider sensor's mobility, and they typically disseminate queries to all the sensors in the WSN. However, queries only need to be disseminated to mobile sensors in our scenario.

A direct method is every stationary sensor node broadcasts the queries to mobile sensors, which wastes precious energy and time. In our approach, a base station assigns each incoming query a unique qid and converts it into the *query item* (qid, predicate, sample period, duration, attribute list, aggregation operator, radius). It disseminates the query item to each mobile sensor along the mobile sensor route in QA-tree to avoid frequent broadcasting. Note that only the sensors in the mobile sensor route need to unicast the query item to the next hop to mobile sensors.

##### 3.1.2 Event detection

As mentioned in Section 1, efficient determination of query areas is a key challenge. We distribute event detection to each mobile sensor to save the communication cost. After receiving a query item, each mobile sensor keeps sensing the required data every sample period to detect the event that satisfies the query predicate for a specific duration. When the event is detected, mobile sensors become reference sensors and determine the query area locally with their current locations as the center of the query area.

##### 3.1.3 Query propagation

In a typical aggregation query processing, queries are only disseminated once from a base station to all the sensors. However, in ETLAQ processing, each reference sensor propagates the triggered query including its current location and the query item to the stationary sensors in the query area.

Each reference sensor broadcasts the triggered query to neighbor stationary sensors. Consider a stationary sensor node  $S$  that receives the triggered query. If  $S$  is a leaf node, it forwards the triggered query to its parent node, because its siblings may not be in  $S$ 's radio range thus do not receive the query from  $S$  though they are in the query area. If  $S$  is not a leaf node, it forwards the triggered query to both its parent node and some of its child nodes. These child nodes are out of reference sensor's radio range and their MBRs overlap the query area. It is easy to prove that if a triggered query is received by at least one sensor node, it will be propagated to all the sensors in the query area. In-network query propagation reduces the transmission cost and saves time compared to the centralized approach in which a reference sensor sends the query back to base station which then disseminates the query to the sensors in query areas.

### 3.1.4 Dynamic in-network aggregation

On receiving the triggered query, leaf sensor nodes sense and transmit the data required in the triggered query immediately, while non-leaf sensor nodes sense the data and wait for the data from their neighbor nodes in the query area before transmission. In previous aggregation approaches [1, 6, 9, 10], sensor nodes always transmit the data to their parent nodes. 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. In our approach, each 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 the sensor. Otherwise, each sensor node adaptively transmits the data to the neighbor node with the highest level in the query area. As a result, few sensor nodes are involved in the aggregation and more data are aggregated locally, so transmission cost can be reduced.

Each non-leaf node starts to aggregate its data with the data from child nodes locally after it receives all the child nodes' messages. If some of its child nodes choose another node to transmit data, it also receives the messages due to the broadcast nature of sensor transmission and will not wait for these nodes' data. To deal with loss of some child nodes' messages caused by nodes' failure or other reasons, a wait interval is assigned for each non-leaf nodes. We assume maximal depth of QA-tree is  $d$ , non-leaf sensor node  $S$  is at level  $l$ , each sensor's sensing and processing time is  $SPT$  and transmission time is  $TMT$ . The lower bound of  $S$ 's wait interval  $WI_s$  satisfies:

$$WI_s = (d - l)(2 \times TMT + SPT)$$

$WI_s$  takes the longest transmission hops  $d - l$  into account that  $S$  forwards the triggered query to the leaf nodes in the query area and leaf nodes' data are transmitted back to  $S$ . If  $WI_s$  expires,  $S$  starts to perform aggregation without waiting for all the child nodes' messages.

After aggregation, each sensor node forwards the aggregation result to the neighbor node that satisfies above conditions, piggybacking the number of sensors in the aggregation. If there are more than one such neighbor nodes, the tie is solved by choosing the neighbor node that has more child nodes in the query area, because this node may aggregate more sensors' data locally.  $S$  sends a message to neighbor nodes to request the number of their child nodes in the query area. It is possible that  $S$  and several other sensor nodes transmit data to each other at the same time. To avoid the collisions, we add to each sensor node's wait interval with an extra wait time that is proportional to the number of its child nodes in the query area, which enforces the sensor node having more child nodes in the query area to wait for other nodes' data. Suppose each non-leaf node has at most  $n$  child nodes in QA-tree and  $S$  has  $i$  child nodes in the query area ( $0 \leq i \leq n$ ),  $S$ 's wait interval  $WI_s$  is updated to:

$$WI_s = (d - l)(2 \times TMT + SPT) + i \times TMT$$

In summary, each non-leaf sensor node performs dynamic in-network aggregation according to the steps in Figure 3.

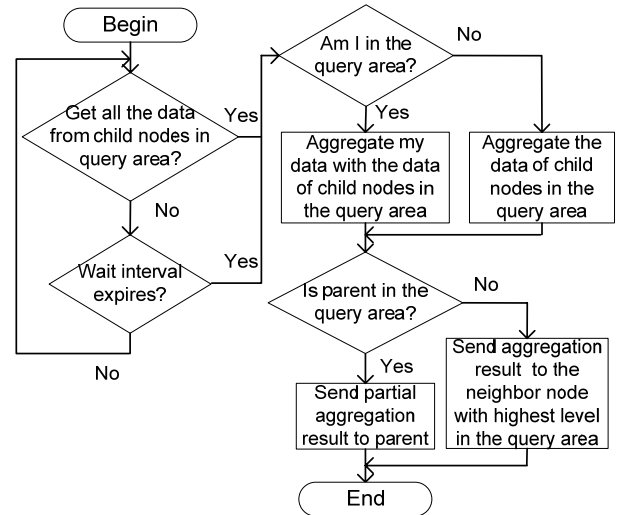


Figure 3. Dynamic in-network aggregation.

## 3.2 Multi-Query Optimization

Existing multi-query optimization schemes [1, 5] do not consider ETLAQs' unique properties such as mobility of sensors and continuously changing query areas. We present an on-line query insertion algorithm that inserts the incoming query  $Q_i$  into a qualified synthetic query  $Q_{syn}$  to maximize sharing between them.

An ETLAQ  $Q_i$  is decomposed into five parts: attribute set  $Q_i.att$ , predicate  $Q_i.p$ , sample period  $Q_i.sp$ , radius  $Q_i.r$  and duration  $Q_i.d$ . By comparing each part with synthetic query  $Q_{syn}$ , our algorithm decides whether  $Q_i$  can be inserted into  $Q_{syn}$  using *hidden rules* and *compatible rules*. Hidden rules for ETLAQ are: H(1)  $Q_i.att \subseteq Q_{syn.att}$ , H(2)  $Q_i.p \subseteq Q_{syn.p}$ ,

H(3)  $Q_i.r \leq Q_{syn}.r$ , H(4)  $Q_i.sp \mid Q_{syn}.sp$ , H(5)  $Q_i.d \leq \text{remaining } Q_{syn}.d$ . H(4) means  $Q_i$ 's sample period is divided by  $Q_{syn}$ 's sample period. If  $Q_i$  satisfies all the hidden rules, it does not need to be injected to the WSN, because its result set is totally included in  $Q_{syn}$ 's result set. Hidden rules are strict for many queries and do not exploit the similarities among queries at a fine granularity. We loose hidden rules to *compatible rules*: C(1)  $Q_i.p \supset Q_{syn}.p$ , C(2)  $Q_i.r > Q_{syn}.r$ , C(3)  $Q_{syn}.sp \mid Q_i.sp$ , C(4)  $Q_i.d > \text{remaining } Q_{syn}.d$ .

When an incoming query  $Q_i$  arrives at a base station, our algorithm first scans the synthetic query list. If  $Q_i$  is found to satisfy all the hidden rules with any synthetic query,  $Q_i$  does not need to be injected to the WSN. If such synthetic query does not exist, the algorithm then tries to insert  $Q_i$  into a compatible synthetic query to form  $Q'_{syn}$  as the following steps:  $Q'_{syn}.att$  is updated to  $Q_i.att \cup Q_{syn}.att$ . For other four parts, if some parts of  $Q_i$  and  $Q_{syn}$  satisfy compatible rules and others satisfy hidden rules,  $Q_i$  and  $Q_{syn}$  are compatible and can be rewritten into a new synthetic query  $Q'_{syn}$ . The values of  $Q_{syn}$ 's parts that satisfy compatible rules with  $Q_i$  are updated by the value of the same parts in  $Q_i$ .  $Q_{syn}$ 's remaining parts are not changed. If more than one compatible synthetic query is found in the list, the synthetic query that has the most parts satisfying hidden rules with  $Q_i$  is chosen, because they have most sharing among sensor nodes.  $Q'_{syn}$  is disseminated to each mobile sensor to replace  $Q_{syn}$ . If there is no qualified synthetic query available,  $Q_i$  is added to synthetic query list and disseminated to mobile sensors.

#### 4. RELATED WORK

Location queries on moving objects have been studied in both centralized and distributed system [2, 3]. However, there is little work found in the literature associate with location aware data services in mobile WSNs.

In [9], *Tiny Aggregation Service (TAS)* for WSNs is presented which can be used to answer simple and declarative aggregation queries. Several techniques such as pipelined aggregate, shared channel and hypothesis testing are proposed in [10] to improve the reliability and performance of aggregation queries processing presented in [9]. Some interesting problems on *acquisitional query processing (ACQP)* in sensor networks are identified in [4] based on TinyDB. In a recent work [1], the authors propose a *Two-Phase Self-Join (TPSJ)* scheme to efficiently evaluate self-join query for event detection in sensor networks. However, mobility is not considered in all the above works. ZebraNet [8] proposes an energy-efficient scheme for tracking mobile sensors; however, it does not refer to processing aggregation query.

A *Two-Tier Multiple Query Optimization (TTMQO)* scheme is proposed in [5]. The base station groups incoming

queries into synthetic queries based on the cost model to save the total cost of query processing. However, it only considers simple aggregation queries.

#### 5. CONCLUSIONS

In this paper, we have proposed an event detection and in-network aggregation strategy to provide event-triggered location aware data services in mobile WSNs. Such data service query type exists in a wide range of data centric applications especially in the area of the human environment interaction. Our strategy can also be efficiently applied to typical query types in WSNs. We are in the process of improving the reliability of our system to deal with node failure and message losses. In the future, we plan to extend our strategy to support more complex event detection.

#### 6. REFERENCES

- [1] X. Yang, H. B. Lim, M. T. Ozsu, and K. L. Tan. In-Network Execution of Monitoring Queries in Sensor Networks. In *Proceeding of SIGMOD*, 2007.
- [2] S. Ilarri, E. Mena, and A. Illarramendi. Location-Dependent Queries in Mobile Contexts: Distributed Processing Using Mobile Agents. *IEEE Transactions on Mobile Computing*, 5, 8 (2006), 1029-1043.
- [3] 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*, 18, 5 (2006), 651-668.
- [4] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM Transaction on Database Systems*, 30, 5 (2005), 122-173.
- [5] S. Xiang, H. B. Lim, K.-L. Tan, and Y. Zhou. Two-Tier Multiple Query Optimization for Sensor Networks. In *Proceeding of ICDCS*, 2007.
- [6] Y. Yao and J. Gehrke, "Query Processing for Sensor Networks," In *Proceeding of CIDR*, 2003.
- [7] A. Soheili, V. Kalogeraki, and D. Gunopulos. Spatial Queries in Sensor Networks. In *Proceeding of GIS*, 2005.
- [8] P. Juang, H. Oki, Y. Wang, et al. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *Proceedings of the ASPLOS-X*, 2002.
- [9] S. Madden, M. J. Franklin, J. M. Hellerstein, and Wei Hong. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *Proceeding of OSDI*, 2002.
- [10] S. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. In *the Workshop on Mobile Computing and Systems Applications*, 2002.