

# EASE: An Energy-Efficient In-Network Storage Scheme for Object Tracking in Sensor Networks

Jianliang Xu

Department of Computer Science  
Hong Kong Baptist University  
Hong Kong  
xujl@comp.hkbu.edu.hk

Xueyan Tang

School of Computer Engineering  
Nanyang Technological University  
Singapore  
asxytang@ntu.edu.sg

Wang-Chien Lee

Department of Computer Science & Engineering  
Penn State University  
University Park, PA  
wlee@cse.psu.edu

**Abstract**—Energy efficiency is one of the most critical issues in the design of wireless sensor networks. Observing that many sensor applications for object tracking can tolerate a certain degree of imprecision in location data of tracked objects, this paper studies precision-constrained approximate queries that trade answer precision for energy efficiency. We develop an Energy-conserving Approximate StorageE (EASE) scheme to efficiently answer approximate location queries by keeping error-bounded imprecise location data at some designated storage node. The data impreciseness is captured by a system parameter, i.e., approximation radius. We analyze the performance of EASE in terms of message complexity and derive the optimal setting of approximation radius. We show via extensive simulation experiments that, as compared to a conventional approach, the EASE scheme cuts down the network traffic by up to 96% and, in most cases, prolongs the network lifetime by a factor of 2–5.

## I. INTRODUCTION

With the rapid development of wireless communications and electronics technologies [1], [2], wireless sensor networks have emerged as a promising solution for a wide range of civil and military applications [3]–[8]. A sensor network is constructed of a large number of tiny sensor nodes scattered in an area of interest. These sensor nodes are equipped with data processing, sensing, and communication capabilities. They are usually powered by battery. However, replacing battery is not only costly but impossible in many situations (e.g., in a hard-to-reach area). Thus, energy efficiency is a critical consideration in the design of large-scale sensor networks.

In this paper, we consider object tracking sensor networks, one of the most important classes of sensor networks. Example applications of object tracking include wildlife animal monitoring in remote areas and intrusion detection in military sites. Users in these applications are interested in *location queries*, which return locations of tracked moving objects. There have been significant research efforts towards energy-conserving object tracking sensor networks (e.g., [9]–[12], see Section II for a detailed discussion). Most of them aimed at reducing the number of sensing nodes activated for tracking an object and/or reducing the location updating traffic in providing *accurate* answers to location queries.<sup>1</sup>

<sup>1</sup>The accuracy is based on best effort since the object location cannot be 100% accurate due to network delays and discrete sampling instances, etc.

Imprecision is an inherent property of object tracking sensor networks. The state-of-the-art location positioning technologies such as GPS and triangulation are not error-free. Moreover, due to data transfer delay and constant object mobility, it is almost impossible for a user to obtain the precise position of an object. In addition, many applications are willing to tolerate a certain degree of imprecision or error in data due to either the application nature or the high resource constraints in sensor networks. As such, here we take a different approach to improve energy efficiency by exploiting the tradeoff between data quality and energy conservation. Instead of always feeding the most accurate answers to location queries, we investigate the problem of providing precision-constrained approximate locations based on user tolerances. In our model, an *approximate location query* is specified by an object identifier and a precision constraint. The sensor network responds with a location bounded by the required precision.

In this paper, we develop an Energy-conserving Approximate StorageE (EASE) scheme to efficiently answer approximate location queries. Most existing studies assumed centralized/designated storage for data collection and query answering [9], [13], [14]. In contrast, EASE innovatively maintains two versions of object location data in the network. High-precision data is kept at some *local storage node* close to a moving object in order to reduce long-distance traffic resulted from *remote updates*. Meanwhile, the same data with a lower precision is replicated at some *designated storage node* which is known to users in order to reduce the querying traffic. Within the EASE scheme, the imprecision of location data at the designated storage node is bounded by an *approximation radius*, which specifies a geographical area in which the low-precision location data is considered to be valid. In other words, a location update due to object movement will not be sent to the designated storage node if the object remains within the approximation radius. Correspondingly, a query is answered by the designated storage node if its precision constraint is weaker than that specified by the approximation radius. Otherwise, the query is forwarded to the local storage node for resolution. As such, the EASE scheme optimizes the network performance (in terms of reducing network traffic and energy consumption) by reducing both the updating and

querying traffic. This is achieved by properly setting the approximation radius in order to minimize the overall traffic. The optimal setting of approximation radius is mathematically derived. We show via extensive simulation experiments that the EASE scheme with the optimal setting of approximation radius reduces the network traffic by up to 96% from a conventional approach and that, in most cases, prolongs the network lifetime by a factor of 2–5.

We summarize the contributions made in this paper as follows:

- To the best of our knowledge, this is the first study on data dissemination of object tracking sensor networks that attempts to address energy efficiency by exploiting the trade-off between data quality and energy conservation.
- An energy-efficient in-network storage scheme, called EASE, is proposed to efficiently answer precision-constrained approximate location queries.
- The setting of the proposed storage scheme is analyzed and optimized through a theoretical study.
- An extensive performance evaluation is conducted to evaluate the performance of the proposed EASE scheme. The results demonstrated the superior performance of EASE over the existing approach.

The rest of this paper is organized as follows. Section II reviews the related work. The system model is described in Section III. Section IV presents the proposed EASE scheme in detail. We analyze the performance of EASE and derive the optimal setting of approximation radius in Section V. A performance evaluation is reported in Section VI. Finally, Section VII concludes the paper.

## II. RELATED WORK

Energy efficiency is a major concern of sensor networks. There are two research directions in improving the lifetime of an object tracking sensor network. One is to reduce energy consumption in the sensing component [11], [15]. The basic idea is to activate only the essential sensor nodes needed to track the moving objects while leaving the other nodes in a power-saving mode. In [16], [17], the sensor nodes are organized into a cluster-based architecture such that a cluster head calculates object location based on signal readings from its slave nodes. Based on these work, we assume object location can be obtained in certain detecting cluster head and only focus the task on where and how to store the location data in support of energy-efficient approximate location queries.

The other direction, aligned with ours, is to improve energy efficiency by reducing network traffic in disseminating location updates. Goel and Imielinski [9] proposed a prediction-based monitoring paradigm. A base station collects sensor readings and periodically generates predictions to be sent back to the sensor nodes. A sensor node reports a location update only when its reading differs from the predicted. Xu *et al.* [13] suggested a dual-prediction scheme where a fixed prediction model is deployed at both the base station and the sensor nodes. Kung and Vlah [14] observed the existence

of movement locality and proposed a publish-and-subscribe tracking method to reduce the updating traffic. While there has been research on the trade-off between energy conservation and quality of tracking [10], [18], the trade-off has not been investigated in the dissemination of location data, which is the topic of this paper.

Data storage models have been extensively studied for sensor networks. In particular, in-network storage is advocated by most of the recent research proposals. In the TinyDB project, Madden *et al.* [19] presented pull-based acquisitional query processing (ACQP), where sensors control where, when, and how often data is acquired and delivered to query processing operators. The Cougar project [20] employed a hybrid pull-push model, in which sensed data is pushed to some selected view nodes, from where the data can be pulled when queries are issued. Ratnasamy *et al.* [21] proposed an in-network data-centric storage model: the sensed data is pushed to the sensor node nearest to some geographical location hashed from a predefined key. Only equality queries are supported by the data-centric storage. Zhang *et al.* [22] suggested storing sensed data locally. A centric ring-based index was proposed to facilitate query processing. Liu *et al.* [23] recently developed a “comb-needle” model to support one-shot queries in the sensor network. Unfortunately, none of these previous work has studied approximate data storage to improve energy efficiency.

Han *et al.* [24] is a pioneer to study approximate queries in sensor networks. They developed an efficient data collection protocol to fulfill the application-specified data quality while minimizing the energy consumption of sensor nodes. However, the solutions developed in [24] are not applicable to object tracking applications due to several reasons. First, [24] considered a simple single-hop system where all sensor nodes can send data directly to the server. In contrast, we consider a multi-hop network, which relies on routing protocols to relay data among sensor nodes. Data dissemination in a multi-hop system is much more complicated than that in a single-hop system. Second, [24] assumed that the phenomenon being sensed is stationary, whereas we are interested in tracking moving objects which change their locations (and hence local storage nodes) from time to time. Moreover, the cost of location updates/queries depends on the location of the moving object. These above differences make our system modeling and performance analysis completely different from [24]. Precision-constrained queries have also been studied for in-network data aggregation [25], which has a different focus from object tracking sensor networks.

## III. SYSTEM MODEL

We consider a sensor network consisting of a large number of stationary sensor nodes deployed in some operational area. Each sensor node is aware of its own location, through GPS for example. We assume that the nodes organize themselves into clusters and that every cluster has a cluster head. A cluster head is more powerful than an ordinary sensor node. It is equipped with some local storage to store data, and is

also capable of communicating with other cluster heads to exchange data. The sensor nodes in a cluster can work together to recognize and track the objects in their vicinity, e.g., a cluster head can determine object location based on signal readings from its slave nodes [16], [17]. Location tracking is an important area of research in the sensor applications, but out of scope of our study. We also assume that the moving objects being tracked are identifiable (i.e., a unique object identifier is assigned to each object). Since this paper focuses on the issues of energy-efficiently storing and disseminating object location data in support of approximate location queries rather than the object recognition and location tracking techniques, we do not consider the energy cost of sensing, data fusion, and object tracking in this paper. We shall focus ourselves on communication amongst cluster heads only. Unless explicitly specified, a sensor node refers to a cluster head for the rest of this paper.

**Approximate Location Queries.** The sensor network under consideration supports a large number of users making one-shot queries for the locations of moving objects. The queries can be made via a sensor node (known as *querying node*) from anywhere in the network (e.g., from the mobile devices such as PDAs held by the users). Each approximate location query is specified by a tuple  $\langle object\_id, p \rangle$ , where *object\_id* is the identifier of the target object, and *p* is the precision constraint in object location that the query can tolerate.<sup>2</sup>

**Centralized Storage (CS) scheme.** The sensor nodes sample and decide the objects' locations at a fixed *sampling rate*. Intuitively, the object location data can be stored at 1) the sensor node (cluster head) detecting the object; or 2) a centralized storage node. If the location data is stored locally at a detecting node, a query that wants to find the location of some object has to *flood* the whole network. Obviously, this scheme is not cost efficient. In contrast, if the centralized storage (CS for short) scheme is adopted in the system, any location update of a moving object is sent to some *designated storage node*, which could be a centralized base station with the external storage or an in-network sensor node with the data centric storage (DCS) [21] (see Figure 1(a)).<sup>3</sup> When a query is issued for an object, it is sent to the corresponding centric storage node, which would return the result to the querying node. However, the high updating traffic incurred is a major deficiency of the CS scheme. In the next section, we are going to propose a hybrid in-network storage scheme that achieves the best overall performance by balancing the querying cost and updating cost for approximate location queries. Without loss of generality, we shall assume for the rest of this paper that

<sup>2</sup>For simplicity, we assume in this paper that *p* is the error tolerance in addition to the unavoidable system error such as inaccuracy of the positioning technique. In practice, *p* can be determined by the user's error tolerance minus the maximum system error.

<sup>3</sup>The centric storage node of an object is determined by applying a pre-defined hash function  $H(\cdot)$  (e.g., Geographic Hash Table (GHT) [21]) on the object identifier. Thus, the centric storage nodes might be different for different objects.

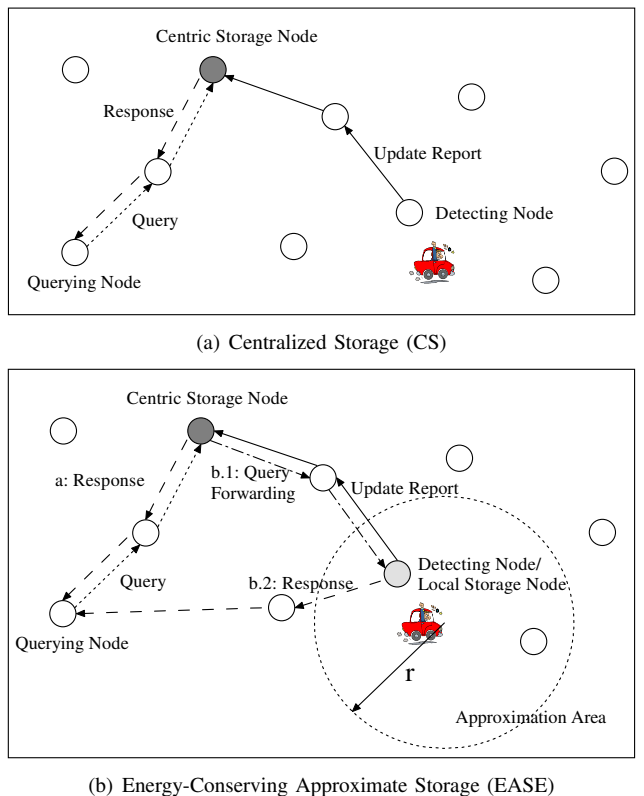


Fig. 1. Illustration of Query Processing Procedure

the in-network DCS [21] is employed for centralized storage.

#### IV. EASE: ENERGY-CONSERVING APPROXIMATE STORAGE

This section proposes an *Energy-conserving Approximate StorageE* (EASE) scheme that takes advantage of the error tolerances of queries. We first give an overview of the EASE scheme in Section IV-A. Section IV-B describes the location updating protocol working with EASE. Finally, we discuss how EASE handles node failures and message losses in Section IV-C.

##### A. Overview

Consider a moving object. Since approximate queries do not always require high-precision location data, the EASE scheme attempts to cut down the updating traffic by storing an imprecise version of the object location data at the centric storage node. Specifically, EASE stores an approximate location bounded by a certain error at the centric storage node. The error bound is represented by an *approximation radius*. A stored location *o* with an approximation radius of *r* means that the object must be in an *approximation area* defined by a circle with *o* as the center and *r* as the radius. The high-precision location data of the tracked object is kept at a *local storage node* close to the object's current position (to be defined in the next subsection). The local storage node, dynamically adapting

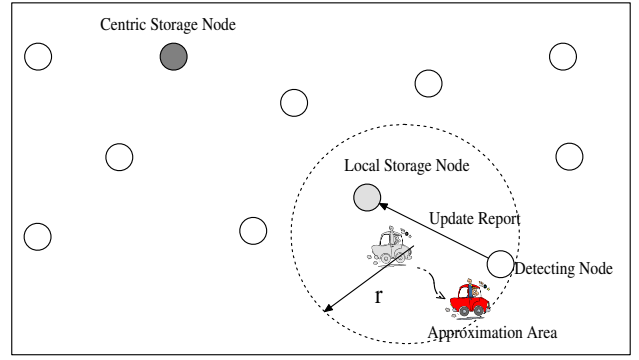
to the object mobility, aims at reducing the cost of remote updates.

In the proposed EASE scheme, a centric storage node plays two important roles: 1) a storage server of low-precision location data for approximate queries; and 2) an index server that keeps track of the local storage nodes (without this information, the local storage nodes can only be found by flooding). Thus, a centric storage node maintains two items for an object: an approximate location of the object and the local storage node of the object. The former is used to answer the queries with less stringent precision constraints while the latter is used to forward the queries with more stringent precision constraints. More specifically, upon receiving an approximate location query with a precision constraint  $p$ , the centric storage node checks the current approximation radius  $r$  of the stored location. If  $p \geq r$ , the stored location satisfies the precision requirement and, hence, it is returned to the querying node immediately (as shown by Case *a* in Figure 1(b)). If  $p < r$ , the stored location is inadequate in precision. Consequently, the query is forwarded to the local storage node and the result is returned from that node (as shown by Case *b* in Figure 1(b)).

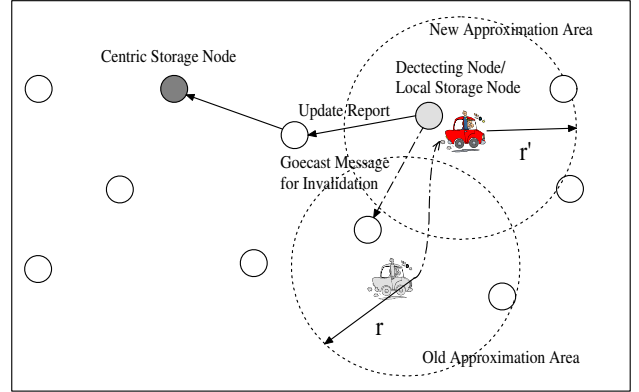
### B. Location Updating Protocol

As discussed above, there are two storage nodes associated with a moving object, i.e., a fixed centric storage node and a dynamically changing local storage node. The two items (i.e., approximate object location and local storage node) maintained by the fixed centric storage node are updated independently. The approximate location needs to be updated whenever the object moves out of the approximation area (defined in the last subsection). The local storage node, on the other hand, is associated with a *coverage area*. All sensor nodes in the coverage area are aware of the local storage node and send location updates of the object to it at sampling instances. When the object moves beyond the coverage area, a new local storage node is selected and, hence, the item maintained by the centric storage node has to be updated. To reduce network traffic, we define the coverage area the same as the approximation area so that the updates to the two items at the centric storage node can be made in a single message. Thus, at each sampling instance, if the detected object location is within its current approximation area, the detecting node sends a *local update* to the local storage node as shown in Figure 2(a). Otherwise, if the object moves out of the current approximation area, the detecting sensor node becomes the *new local storage node*. It sends a *remote update* to the centric storage node (see Figures 2(b) and 2(c)).

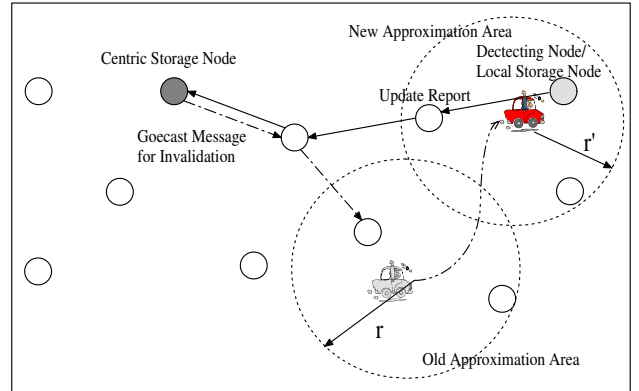
Upon receiving a remote update, the centric storage node updates the location as well as the local storage node of the object. If the approximation radius is updated dynamically, the centric storage node calculates a new radius  $r'$  and returns it to the new local storage node; otherwise the default radius is used as the new radius  $r'$ . A new approximation area is formed as a circle centered at the object's current location. To notify the sensor nodes in this area of the new local



(a) Local Update



(b) Remote Update: Case I



(c) Remote Update: Case II

Fig. 2. Location Update Dissemination with EASE

storage node, the detecting node sends out a *geocast* message (see below for how it works) [27]. If the detecting sensor node is *covered* by the previous local storage node and thus knows the obsolete approximation area,<sup>4</sup> it also sends another geocast message to the sensor nodes in the obsolete approximation area to invalidate the recorded local storage node (see Figure 2(b)). Otherwise, if the detecting node is not aware of the previous local storage node, it informs the

<sup>4</sup>The sensor nodes covered by a local storage node include all nodes inside the approximation (coverage) area as well as those immediate neighbors; this will become more clear when we discuss geocasting in the next paragraph.

centric storage node about this in the remote update and asks the centric node to invalidate the obsolete approximation area using geocast (see Figure 2(c)).

A geocast message takes the parameters of a geographical area to receive the message and the type of action (either *notification* of a new approximation area or *invalidation* of the obsolete approximation area). For notification geocast, the identifier of the new local storage node is also included in the message. A geocast works in two phases. In the first phase, the message is routed towards the geographical area using some geographical routing protocol such as GPSR [26]. If the current sensor node is already within the geographical area, the first phase is not needed. After reaching the target geographical area, the message is flooded to all sensor nodes in the area through broadcast. When a sensor node receives such a broadcast message, if it is within the target geographical area, the sensor node further broadcasts the message to all its neighbors. Otherwise, if the receiving sensor node is outside the target geographical area, no further action is taken. In this way, a geocast message will be received by all the nodes within the geographical area as well as their immediate neighbors.

### C. Discussion

Wireless sensor networks are unreliable and error-prone in nature. In this section, we discuss how EASE deals with node failures and message losses.

The EASE scheme relies on a centralized base station or data centric storage (DCS) to store imprecise location data. To address reliability and availability, DCS enhances the centric storage nodes by replication [21], [28]. EASE stores high-precision location data at local storage nodes, which can be replicated for performance enhancement using similar techniques employed by DCS [21], [28].

Due to a high cost of reliable communication, message routing and transmission often work in a best-effort manner for sensor networks. If a message is lost during transmission, it is costly to recover it at the networking layer. Nevertheless, we can remedy it at the application layer. For query messages, we set a *patience* value (i.e., twice of the round trip time between two farthest nodes). If a query is not answered within the patience time, the user application may assume the previous query has lost in the network and resend the query. For update messages, we need not do anything for message losses since this only introduces a little imprecision in object locations. Note that the problem of query and update message losses is faced by any data dissemination schemes including CS and EASE.

There are two types of geocast messages (i.e., invalidation and notification) specific to EASE. If a notification message is lost, the sensor network still functions properly. In the worst case, if a sensor node is not notified of the local storage node, upon detecting the object, it will send a remote update to the centric storage node (and become the new local storage node) rather than sending the update to the local storage node. However, failing to invalidate an obsolete approximation area

Notation	Description
$n$	number of sensor nodes
$\lambda$	query rate
$\mu$	sensor sampling rate
$C_{qn}$	cost of answering a query by centric storage
$C_{qf}$	cost of answering a query by local storage
$C_{ur}$	cost of a remote update
$C_{ul}$	cost of a local update
$r$	approximation radius, i.e., the error bound
$f$	sensor node density
$p_m$	maximum imprecision
$C(r)$	overall complexity with approximation radius of $r$
$p_{qf}(r)$	probability of not being satisfied by centric storage
$\eta(r)$	remote update rate
$d$	moving distance of each step in random walk
$l$	time taken to move a single step in random walk

TABLE I  
NOTATIONS USED IN ANALYSIS

might lead to an improper function of the sensor network. A detecting node in the obsolete area may falsely send updates to the local storage node that has been replaced by some other node. To handle this issue, we assign an *expiration time* to the local storage node. In a perfect network, the local storage node would receive a local update at every sampling instance. As such, if the local storage node has not received any update for a period of expiration time, which is set at three times of the sampling interval, it assumes that some other node has taken up the local storage and broadcasts an invalidation message to its approximation area. Note that such an approach may cause a false operation. For example, in case all of three consecutive local update messages are lost, the local storage node will regard this as an indication of invalidation message loss and incorrectly invalidate the *valid* approximation area. Fortunately, such a chance is very small. Moreover, the sensor network can recover from the false operation by making the next detecting node as the new local storage node.

## V. PERFORMANCE ANALYSIS AND OPTIMIZATION

In this section, we first analyze the performance of EASE in terms of message complexity. Then, we derive the optimal setting of approximate radius for EASE.

### A. Performance Analysis

This section analyzes the message complexity of EASE (i.e., the total number of message transfers observed in the network). A summary of the notations used in the analysis is provided in Table I. For simplicity, data transmission is assumed to be error-free. Assume that the query rate is  $\lambda$  and the sensor sampling rate is  $\mu$ . Let  $C_{qn}$  and  $C_{qf}$  be the costs of answering a query by the centric storage and the local storage respectively, and  $C_{ur}$  and  $C_{ul}$  be the costs of a remote update and a local update respectively. Given an approximation radius of  $r$ , it is easy to see that the overall message complexity is given by

$$C(r) = (1 - p_{qf}(r)) \cdot \lambda \cdot C_{qn} + p_{qf}(r) \cdot \lambda \cdot C_{qf} + \eta(r) \cdot C_{ur} + \mu \cdot C_{ul}, \quad (1)$$

where  $p_{qf}(r)$  is the probability that the approximate location stored at the centric storage does not satisfy the query specific precision and  $\eta(r)$  is the rate of remote updates (i.e., the rate of moving out of the approximation area). The four terms in the above formula represent the cost incurred by the queries answered by the centric storage node, the cost incurred by the queries answered by the local storage node, the cost of remote updates, and the cost of local updates respectively.

Assume that the sensor network consists of  $n$  uniformly distributed sensor nodes. As in [21], [22], we use  $n$  to approximate the cost of flooding a message to the whole network, and use  $\sqrt{n}$  to approximate the cost of sending a message between two nodes in the network. It is easy to see (from Figure 1(b)) that

$$C_{qn} = 2\sqrt{n}, \quad (2)$$

$$C_{qf} = 3\sqrt{n}. \quad (3)$$

Note that the costs of  $C_{ur}$  and  $C_{ul}$  are a function of approximation radius  $r$ . We now derive these two costs and  $p_{qf}(r)$ . The rate of remote updates  $\eta(r)$  depends on the mobility pattern, and will be discussed in the next section.

A remote update involves sending the update to the centric storage and two geocast messages to notify the new approximation area and invalidate the obsolete approximation area respectively. The remote update cost is approximated by  $\sqrt{n}$ . Since a geocast message is sent to an approximation area which is usually nearby; for simplicity, we neglect the routing cost and approximate its cost by the number of sensor nodes in the approximation area, i.e.,  $\pi r^2 f$ , where  $r$  is the approximation radius and  $f$  is the sensor node density. Therefore, we obtain  $C_{ur}$  as

$$C_{ur} = \sqrt{n} + 2\pi r^2 f. \quad (4)$$

For a local location update, the average travel distance is given by:

$$\frac{\int_0^r (x \times 2\pi x) dx}{\int_0^r (2\pi x) dx} = \frac{2/3\pi r^3}{\pi r^2} = \frac{2}{3}r. \quad (5)$$

Thus, the local update cost can be approximated by the average number of sensor nodes encountered when travelling a distance of  $\frac{2}{3}r$ , i.e.,

$$C_{ul} = \frac{2}{3}r\sqrt{f}. \quad (6)$$

We assume that the query precision constraint is uniformly distributed in the range of  $[0, p_m]$ . Thus, the probability of a query not being satisfied by the centric storage is

$$p_{qf}(r) = \min\left\{\frac{r}{p_m}, 1\right\}. \quad (7)$$

Combining (1) through (7), we can rewrite (1) as

$$C(r) = \lambda \cdot 2\sqrt{n} + \min\left\{\frac{r}{p_m}, 1\right\} \cdot \lambda \cdot \sqrt{n} + \eta(r) \cdot (\sqrt{n} + 2\pi r^2 f) + \mu \cdot \frac{2}{3}r\sqrt{f}. \quad (8)$$

As can be seen, the overall message complexity is basically a function of approximation radius  $r$ . The next section discusses the setting of  $r$  in detail.

### B. Optimal Approximation Setting

This section studies the optimal approximation setting based on some known mobility pattern.<sup>5</sup> We assume a 2-dimensional random walk model [14], in which the object moves in steps.<sup>6</sup> It moves a distance of  $d$  along an arbitrary direction (i.e., with angle  $\theta$  uniformly distributed in  $[0, 2\pi)$ ) at each step. Each step takes a duration of  $l$ .

Let  $T(r)$  be the average time an object takes to move out of a circle of radius  $r$ . With a random walk model, we have the following approximation (see the Appendix in [29] for details):

$$T(r) = \left(\frac{r}{d}\right)^2 \cdot l. \quad (9)$$

Therefore, the rate at which the object moves out of the circle (i.e., the approximation area) is given by

$$\eta(r) = \frac{d^2}{lr^2}. \quad (10)$$

Plugging (10) into (8), we get the overall message complexity as follows:

$$C(r) = \lambda \cdot 2\sqrt{n} + \min\left\{\frac{r}{p_m}, 1\right\} \cdot \lambda \cdot \sqrt{n} + \frac{d^2\sqrt{n}}{lr^2} + \frac{2\pi f d^2}{l} + \frac{2}{3}\mu r\sqrt{f}. \quad (11)$$

Let  $\frac{\partial C(r)}{\partial r} = 0$ . We obtain the optimal settings of  $r$  in two cases:

$$r^* = \begin{cases} \sqrt[3]{\frac{6p_m d^2 \sqrt{n}}{l \cdot (3\lambda \sqrt{n} + 2\mu p_m \sqrt{f})}} & \text{if } r \leq p_m, \\ \sqrt[3]{\frac{3d^2 \sqrt{n}}{l\mu \sqrt{f}}} & \text{if } r > p_m. \end{cases}$$

The one producing the lower message complexity  $C(r)$  will be selected as the final setting of  $r$ .

## VI. PERFORMANCE EVALUATION

In this section, we conduct experiments to compare the proposed EASE scheme against the conventional CS scheme (described in Section III). Since the goal of this study is to reduce network traffic and to improve energy efficiency, the first two sets of experiments focus on the volume of messages and the energy consumption incurred in the sensor network. Then, we compare the query latency of the CS and EASE schemes.

<sup>5</sup>When the mobility pattern changes over time, the approximation setting should dynamically adapt to the system workload [29]. We shall present an adaptive approximation setting algorithm in the full version of this paper.

<sup>6</sup>The 2-dimensional random walk model is used in this paper as a case study. The optimization technique presented here is applicable to other mobility patterns as long as their  $\eta(r)$  can be estimated.

### A. Simulation Setup

To facilitate our comparison, we developed a simulator based on ns-2 (version 2.26) [31] and NRL’s sensor network extension [32]. Table II summarizes the system parameters and their settings used in our experiments.

We simulate 225 sensor nodes (i.e., cluster heads) as deployed on a  $500 \times 500\text{-}m^2$  field. The field is divided into  $225 \ 34 \times 34\text{-}m^2$  grid cells, each of which has a sensor node at the center. Each sensor node can detect the objects located in its grid cell and position them [16], [17]. As in the previous work [21], [26], the simulator uses a modified IEEE 802.11 radio with a  $40\text{-}m$  radio range in its MAC layer. The maximum number of retransmissions at the MAC layer is set at 7. The sensor nodes can intelligently switch among three states (with different power consumptions): sending messages, receiving message, and sleeping. The simulator includes an energy model that measures the energy consumption of each sensor node. For simplicity, the energy overhead for state switching is ignored. Queries for the location of a moving object are issued randomly from the sensor nodes in the field. The precision constraints of the queries are uniformly distributed between 0 and  $p_m$ .

As stated in the system model (Section III), this research assumes that sensor nodes are stationary and location-aware. We have also assumed the in-network DCS scheme [21] for centralized storage such that the centric storage node of an object is determined by applying a pre-defined hash function  $H(\cdot)$  on the object identifier. To facilitate geographical message routing (such as for update reporting and query forwarding), the greedy perimeter stateless routing (GPSR) protocol [26] is employed by the simulator. GPSR operates in two modes: greedy mode and perimeter model. In the greedy mode, the forwarding node forwards the message to the neighbor nearest to the destination. If no such neighbors exist, the algorithm switches to the perimeter mode, which recovers by routing around the perimeter of the region.

The random walk mobility model is used to simulate the moving pattern of objects in this simulation. Several mobility profiles, including *slow*, *moderate*, and *fast*, are listed in Table III. The optimal settings of approximate radius,  $r$ , can be obtained using the optimization technique described in Section V-B. Table IV shows the optimal settings of  $r$  for slow, moderate, and fast mobility profiles under a number of decreasing precision requirements (denoted by maximum imprecision,  $p_m$ ).

As shown in the table, when the precision requirement is high (i.e.,  $p_m = 0$  for *slow* and *moderate* profiles;  $p_m \leq 20$  for *fast* profile), a large  $r$  ( $> p_m$ ) is chosen in EASE to suppress remote location updates. In such cases, all queries will be forwarded to the local storage nodes for high-precision answers; the selection of a large  $r$  implies that the cost reduction for location updates outweighs the overhead for query forwarding. As the precision requirement gets lower, the EASE algorithm chooses a smaller  $r$  ( $< p_m$ ) to limit query

Parameter	Setting
Field Size	$500 \times 500 \ m^2$
Number of Nodes ( $n$ )	225
Node Density ( $f$ )	1 node / $34 \times 34 \ m^2$
Radio Range	$40 \ m$
Power Consumption in Sending Message	$14.88 \ mW$
Power Consumption in Receiving Message	$12.50 \ mW$
Power Consumption in Sleeping Mode	$0.016 \ mW$
GPSR Beacon Interval	$3 \ sec$
GPSR Beacon Expiration	$13.5 \ sec$
GPSR Implicit Beacon	yes
Sensor Sampling Rate ( $\mu$ )	$5 / sec$
Query Rate ( $\lambda$ )	$0.1 - 10 / sec$
Maximum Imprecision ( $p_m$ )	$10 - 100 \ m$
Query Message Payload Size	10 bytes
Update Message Payload Size	40 bytes
Result Message Payload Size	40 bytes
Geocast Message Payload Size	10 bytes
Query Start Time	$20 \ sec$
Simulation Time	$500 \ sec$

TABLE II  
SYSTEM PARAMETERS AND SETTINGS

Mobility Profile	Step Distance ( $d$ )	Step Interval ( $l$ )
Slow	$1 \ m$	$1 \ sec$
Moderate	$5 \ m$	$1 \ sec$
Fast	$15 \ m$	$1 \ sec$

TABLE III  
MOBILITY PROFILE SETTINGS

forwarding and reduces the overall traffic. As the precision requirement further decreases, the approximation radius  $r$  is then slightly increased to reduce the updating traffic.

The simulation measures the performance in terms of *message complexity* (# message transfers in the network) and *energy consumption* of the sensor nodes. Since this paper focuses on data dissemination, the energy performance measure excludes the sensing component. Similar to [21], we do not measure the message overhead incurred by the underlying routing protocol (e.g., beacons in GPSR). Such overhead is usually of lower order than the application data traffic. To facilitate an easy illustration and comparison of EASE and CS, we use *normalized cost* for the primary metrics (i.e., message complexity and energy consumption). The normalized cost is defined as the ratio of a measured cost of EASE to that of CS. The smaller is the normalized cost, the better does EASE perform. Each simulation run lasted for 500 seconds of simulated time; the first 20 seconds were considered the warm-up period to eliminate the initialization effect.

### B. Message Complexity

This section examines the message complexity of EASE under the optimal approximation setting. Figure 3 shows the message complexity of EASE normalized by that of CS under a number of different maximum imprecision constraints. As observed in the figure, EASE improves the performance over CS by 47.5%-70%. For a reason explained in the last

Max Imprecision ( $p_m$ )	0	10	20	50	80	100
Slow	7.42	2.67	3.32	4.32	4.86	5.12
Moderate	21.7	7.81	9.69	12.6	14.2	15.0
Fast	45.1	45.1	45.1	26.2	29.6	31.1

TABLE IV  
OPTIMAL SETTINGS OF APPROXIMATION RADIUS  $r$

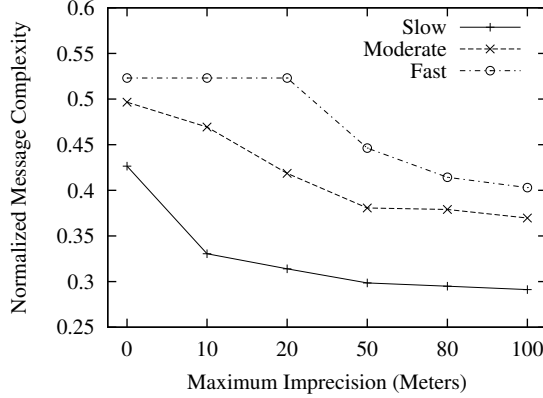


Fig. 3. Message Complexity vs. Precision Requirement ( $p_m$ )

subsection, even when the maximum imprecision is set at zero, EASE still achieves 47.5%-57.5% of improvement. In general, the lower is the precision requirement (i.e., larger maximum imprecision), the more improvement is achieved by EASE. This is expected since EASE is designed to exploit the tolerance of imprecision in object location. A lower precision requirement implies more space for performance improvement. Among the three mobility profiles, the *slow* objects obtain the most significant performance improvement. This is mainly because of the high movement locality exhibited by the *slow* objects; most updating traffic is due to local updates which are propagated to the local storage nodes only.

To gain more insight into how EASE improves the performance, we provide a breakdown of the traffic in Figure 4, where  $p_m$  is set at 50  $m$ . In the CS scheme, a significant

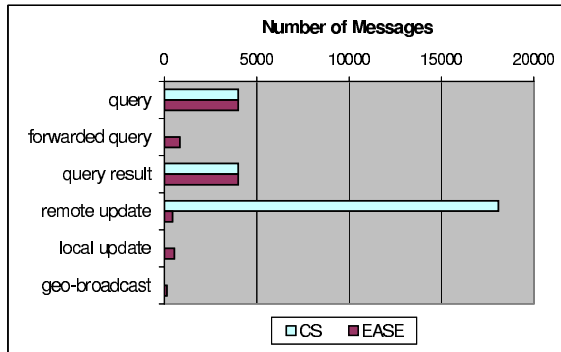


Fig. 4. Breakdown of Message Complexity (Moderate,  $p_m = 50 m$ )

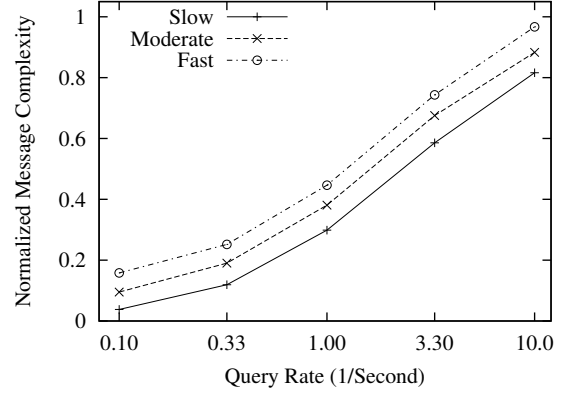


Fig. 5. Message Complexity vs. Query Rate

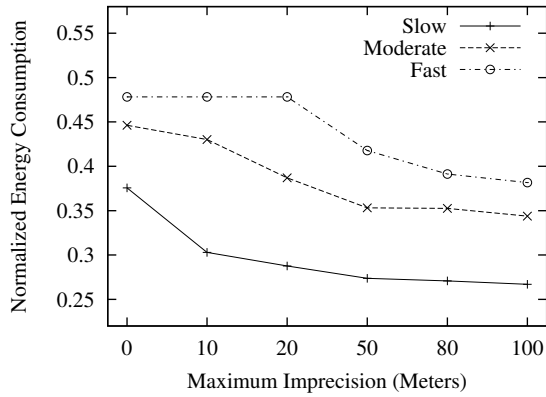
amount of traffic contributes to remote updates. By keeping imprecise data at the centric storage, EASE reduces remote updates by several orders of magnitude. Figure 4 also shows that EASE's overhead for forwarding queries and geocasting is trivial compared to the reduced updating traffic. Overall, EASE reduces the total message complexity over CS by 62.5%.

Figure 5 shows the normalized message complexity of EASE as a function of query rate. When the query rate is low, most network traffic is due to location updates. In this case, EASE significantly cuts down the updating traffic and reduces the overall message complexity by up to 96%. However, when the query rate is extremely high (e.g., 10), queries (and their answers) are the main source of network traffic. In this case, the reduced updating traffic cannot contribute much to the overall message complexity and, hence, EASE has a performance similar to CS.

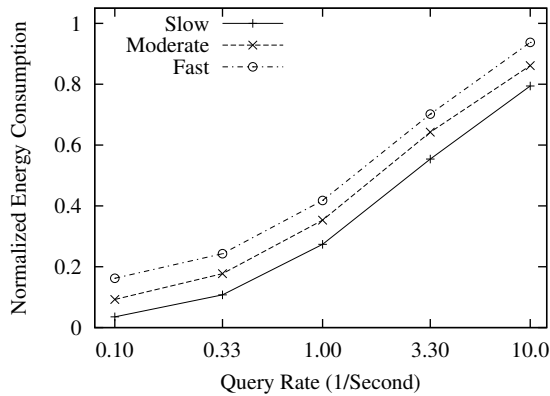
### C. Energy Consumption

We now proceed to evaluate the energy consumption of EASE under the optimal approximation setting. Figure 6 shows the total energy consumed by the sensor network during a simulation run, which is normalized by that of CS. Basically, the trends for energy consumption are similar to those for message complexity (Figures 3 and 5). Nevertheless, it is encouraging to see that the performance improvement is even higher in terms of energy consumption. This is because EASE reduces large-size update messages at the cost of small-size forwarded-query messages and geocast messages. Since more energy is consumed for sending and receiving a larger message, EASE achieves more saving in energy consumption than in message complexity.

We also evaluate the balance of energy consumption in the sensor network and the lifetime of the network by measuring the energy consumption at each individual sensor node. The most energy-consuming node generally determines the lifetime of the sensor network and thus its energy consumption is used as the performance metric in Figure 7. We can see that, in most cases, the energy consumption of EASE is only 20%-50% of that of CS. This implies that EASE can prolong the network



(a) Energy Consumption vs. Precision Requirement



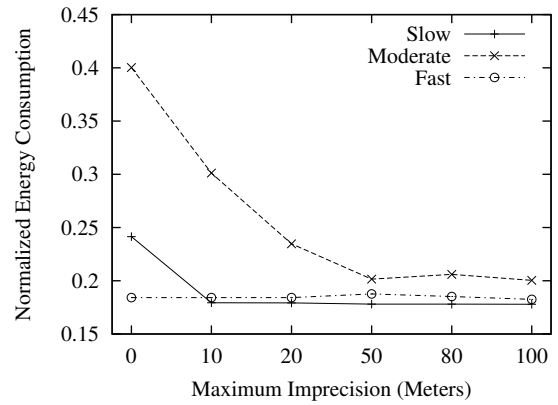
(b) Energy Consumption vs. Query Rate

Fig. 6. Total Energy Consumption

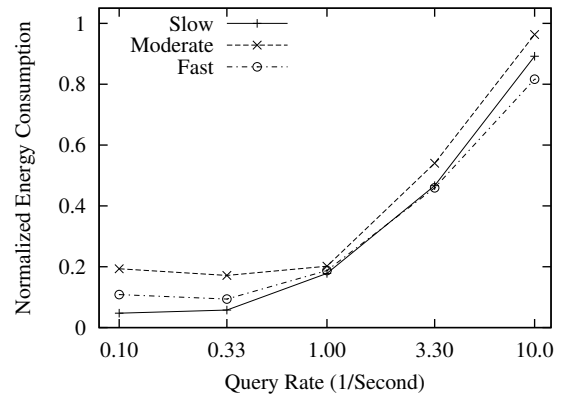
lifetime by a factor of 2–5. It is interesting to note that unlike the case of total energy consumption, EASE achieves a better performance for *fast* objects than *moderate* objects. This is explained as follows. With CS, all queries and updates are sent to the centric storage nodes. Hence, the centric storage nodes are hotspots. However, with reduced updating traffic by EASE, the hotspots are no longer the centric storage nodes. Rather, some local storage nodes become hotspots. On one hand, a faster moving object generates more updating traffic. On the other hand, a faster moving object changes local storage nodes more frequently and, hence, balances the load of the nodes. Combining these effects, EASE has a longer network lifetime for *slow* and *fast* objects than for *moderate* objects.

#### D. Query Latency

The CS scheme is expected to have a very good performance in query latency because the high-precision location data is always available at the centric data node and, thus, a query can quickly be answered there. With the EASE scheme, if a query is not satisfied by the centric storage node due to insufficient precision, it is forwarded to the corresponding local storage node. Thus, the average querying path is lengthened due to such query forwarding. In this section, we measure the average



(a) Most Energy Consumption vs. Precision Requirement



(b) Most Energy Consumption vs. Query Rate

Fig. 7. Energy Consumption of the Most Consuming Node

latency of answering approximate location queries to see how well EASE can achieve the query latency performance of CS.

As shown in Figure 8, when the precision requirement is high (i.e.,  $p_m = 0$  – the worst case), EASE performs worse than CS by no more than 50%. However, as the tolerance of imprecision increases, EASE consistently improves the query latency (and even outperforms CS for large values of  $p_m$ ). This is mainly because for a larger value of  $p_m$ , EASE incurs less network traffic (as shown in Figure 3) and hence less message transmission collisions, which reduces the overall latency even though the querying path is lengthened.

## VII. CONCLUSIONS

This paper presents a study on precision-constrained approximate location queries for object tracking sensor networks. An energy-efficient in-network storage scheme called EASE has been developed to efficiently reduce the network traffic, conserve the energy of sensor nodes, and improve the network lifetime. We have analyzed the optimal setting of approximation radius for EASE. We have also evaluated the proposed techniques through extensive simulation-based experiments. The experimental results demonstrate that the EASE scheme with the derived optimal approximation setting

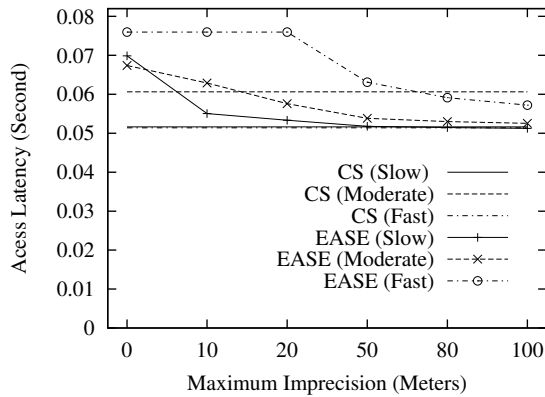


Fig. 8. Query Latency vs. Precision Requirement ( $p_m$ )

saves significant energy consumption for sensor networks and prolongs the network lifetime.

As for future work, we are going to evaluate the performance of EASE under different mobility models [33]. We are interested in incorporating more complex spatial queries (e.g., range queries and k-nearest-neighbor queries) into the approximate storage framework. We also plan to extend the EASE framework to other types of sensor networks, e.g., environmental monitoring sensor networks.

#### ACKNOWLEDGMENT

Jianliang Xu's work was partially supported by grants from the Research Grants Council of the Hong Kong SAR, China (Project No. HKBU 2115/05E and HKBU FRG/04-05/II-26). Xueyan Tang's work was supported in part by a grant from Nanyang Technological University. Wang-Chien Lee's work was supported in part by National Science Foundation grant IIS-0328881.

#### REFERENCES

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramanian, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, August 2002.
- [2] K. Akkaya and M. Younis, "A survey of routing protocols in wireless sensor networks," *Elsevier Journal of Ad-Hoc Networks*, 2004.
- [3] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *ACM MobiCom*, Boston, MA, August 2000.
- [4] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless sensor networks," in *Conference on System Sciences*, Hawaii, Jan. 2000.
- [5] H. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," in *ACM MobiCom*, Atlanta, GA, Sept. 2002.
- [6] S. Bhattacharya, H. Kim, S. Prabh, and T. Abdelzaher, "Energy-conserving data placement and asynchronous multicast in wireless sensor networks," in *ACM MobiSys*, San Francisco, CA, May 2003.
- [7] S. Nath, P. B. Gobbons, S. Seshan, and Z. R. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," in *ACM SenSys*, Baltimore, MD, Nov. 2004.
- [8] N. Shrivastava, C. Buragohain, and D. Agrawal, "Medians and beyond: New aggregation techniques for sensor networks," in *ACM SenSys*, Baltimore, MD, Nov. 2004.
- [9] S. Goel and T. Imielinski, "Prediction-based monitoring in sensor networks: Taking lessons from MPEG," *ACM Computer Communication Review*, vol. 31, no. 5, Oct. 2001.

- [10] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *ACM MobiCom*, Philadelphia, PA, Oct. 2004.
- [11] W. Zhang and G. Cao, "Optimizing tree reconfiguration for mobile target tracking in sensor networks," in *IEEE Infocom*, Hong Kong, March 2004.
- [12] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration for tracking applications," *IEEE Signal Processing Magazine*, March 2002.
- [13] Y. Xu, J. Winter, and W.-C. Lee, "Dual prediction-based reporting mechanism for object tracking sensor networks," in *Annual Conference on Mobile and Ubiquitous Systems (MobiQuitous)*, Boston, MA, August 2004.
- [14] H. T. Kung and D. Vlah, "Efficient location tracking using sensor networks," in *IEEE WCNC*, New Orleans, LA, March 2003.
- [15] J. Aslam, Z. Butler, V. Crespi, G. Cybenko, and D. Rus, "Tracking a moving object with a binary sensor network," in *ACM SenSys*, Los Angeles, CA, Nov. 2003.
- [16] Q. X. Wang, W. P. Chen, R. Zheng, K. Lee, and L. Sha, "Acoustic target tracking using tiny wireless sensor devices," in *Workshop on Information Processing in Sensor Networks (IPSN)*, Palo Alto, CA, April 2003.
- [17] H. Yang and B. Sikdar, "A protocol for tracking mobile targets using sensor networks," in *IEEE Workshop on Sensor Network Protocols and Applications*, Anchorage, AK, May 2003.
- [18] S. Patten, S. Poduri, and B. Krishnamachari, "Energy-quality tradeoffs for target tracking in wireless sensor networks," in *Workshop on Information Processing in Sensor Networks (IPSN)*, Palo Alto, CA, April 2003.
- [19] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *ACM SIGMOD*, San Diego, CA, June 2003.
- [20] A. Demers, J. Gehrke, R. Rajaraman, J. Trigoni, and Y. Yao, "The Cougar project: A work-in-progress report," *ACM SIGMOD Record*, vol. 32, no. 4, Dec. 2003.
- [21] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-centric storage in sensor networks with GHT, a geographic hash table," *ACM/Kluwer MONET*, vol. 8, no. 4, 2003.
- [22] W. Zhang, G. Cao, and T. L. Porta, "Data dissemination with ring-based index for wireless sensor networks," in *IEEE ICNP*, Atlanta, GA, Nov. 2003.
- [23] X. Liu, Q. Huang, and Y. Zhang, "Combs, needles, haystacks: Balancing push and pull for discovery in large-scale sensor networks," in *ACM SenSys*, Baltimore, MD, Nov. 2004.
- [24] Q. Han, S. Mehrotra, and N. Venkatasubramanian, "Energy efficient data collection in distributed sensor environments," in *IEEE ICDCS*, Tokyo, Japan, March 2004.
- [25] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Hierarchical in-network data aggregation with quality guarantees," in *Conference on Extending Database Technology (EDBT)*, Crete, Greece, March 2004.
- [26] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless sensor networks," in *ACM MobiCom*, Boston, MA, August 2000.
- [27] J. C. Navas and T. Imielinski, "GeoCast - geographic addressing and routing," in *ACM/IEEE MobiCom*, 1997.
- [28] A. Ghose, J. Grossklags, and J. Chuang, "Resilient data-centric storage in wireless ad-hoc sensor networks," in *Conference on Mobile Data Management (MDM)*, Jan. 2003.
- [29] J. Xu, X. Tang, and W.-C. Lee, "EASE: An energy-efficient in-network storage scheme for object tracking in sensor networks," Tech. Report, Hong Kong Baptist Univ., April 2005. Available at <http://www.comp.hkbu.edu.hk/~xujl/ease.pdf>.
- [30] C. Olston, B. T. Loo, and J. Widom, "Adaptive precision setting for cached approximate values," in *ACM SIGMOD*, Santa Barbara, CA, May 2001.
- [31] "The network simulator - ns-2," [Online]. Available at <http://www.isi.edu/nsnam/ns/>.
- [32] "NRL's sensor network extension to ns-2," [Online]. Available at <http://nrlsensorsim.pf.itd.navy.mil/>.
- [33] J. Yoon, M. Liu, and B. Noble, "Sound mobility models," in *ACM MobiCom*, San Diego, CA, Sept. 2003.