A New Storage Scheme for Approximate Location Queries in Object Tracking Sensor Networks

Jianliang Xu, *Member, IEEE,* Xueyan Tang, *Member, IEEE,* and Wang-Chien Lee, *Member, IEEE*

***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 the 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 StoragE (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 called the *approximation radius*. We derive the optimal setting of the approximation radius for our storage scheme based on the mobility pattern, and devise an adaptive algorithm to adjust the setting when the mobility pattern is not available a priori or is dynamically changing. Simulation experiments are conducted to validate our theoretical analysis of the optimal approximation setting. The simulation results show that the proposed EASE scheme reduces the network traffic from a conventional approach by up to 96%, and in most cases, prolongs the network lifetime by a factor of 2–5.**

Index Terms: Energy efficiency, data dissemination, data storage, location query, wireless sensor network.

I. Introduction

Owing to the rapid advances in sensing and wireless communication technologies, wireless sensor networks have emerged as a promising solution for a wide range of civil and military applications. 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 on military sites. Users in these applications are interested in *location queries*, which return the locations of tracked moving objects.

A sensor network is typically constructed of a large number of tiny sensor nodes equipped with data processing, sensing, and communication capabilities. The sensor nodes usually operate in an unattended manner and are battery powered. However, replacing the 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. There has been significant research on energy-conserving object tracking sensor networks (e.g., [9], [10], [35]). Most of these studies aimed at reducing the number of sensor nodes activated for tracking an object and/or reducing the location update traffic in providing *accurate* answers to location queries.[1]

Imprecision is an inherent property of object tracking sensor networks. State-of-the-art location positioning technologies such as GPS and triangulation are not error-free. Moreover, 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 trade-off 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.

Inspired by [11] and [23], we develop an Energy-conserving Approximate StoragE (EASE) scheme to efficiently answer approximate location queries. While most prior work assumed centralized/designated storage for data collection and query answering [9], [17], [32], EASE innovatively maintains two versions of object location data in the network. High-precision data are kept at some *local storage node* close to a moving object in order to reduce long-distance traffic resulting from *remote updates*. Meanwhile, the same data with a lower precision are replicated at some *designated storage node* which is known to users in order to reduce the query traffic. In 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 are considered 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 what is specified by the approximation radius. Otherwise, the query is forwarded to the local storage node for resolution. As such, the EASE scheme attempts to optimize the network performance (in terms of reducing network traffic and energy consumption) by balancing the update traffic and query traffic. This is achieved by properly setting the approximation radius. We derive the optimal setting of the approximation

A preliminary report of this work was presented at the Second IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON '05), Santa Clara, CA, September 2005.

Jianliang Xu is with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong. Xueyan Tang is with the School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798. Wang-Chien Lee is with the Department of Computer Science & Engineering, Penn State University, University Park, PA 16802.

[1]The accuracy is achieved by best effort since the object location cannot be 100% accurate due to network delay, discrete sampling instances, etc.

Digital Object Indentifier 10.1109/TPDS.2007.70740

radius based on the mobility pattern, and also devise an adaptive algorithm to adjust the setting on the fly when the mobility pattern is unknown or is dynamically changing. We show via simulation experiments that the EASE scheme, working together with the proposed methods for setting the approximation radius, reduces the network traffic by up to 96% from a conventional approach and, in most cases, prolongs the network lifetime by a factor of 2−5.

We summarize the contributions made in this study as follows:

- To the best of our knowledge, this is the first study on data dissemination in object tracking sensor networks that attempts to address the issue of energy efficiency by exploiting the trade-off between data quality and energy conservation.
- An energy-efficient 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. The theoretical analysis is validated by simulation experiments.
- An adaptive algorithm is proposed to adjust the setting of the approximation radius for unknown/dynamic workloads.
- An extensive performance evaluation is conducted to evaluate the performance of the proposed EASE scheme and the methods for setting the approximation radius.

The rest of this paper is organized as follows. Section II reviews 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 investigate the setting of the approximation radius in Section V. Section VI presents the results of the performance evaluation. Finally, Section VII concludes the paper.

II. Related Work

A. Object Tracking Sensor Networks

There are two research directions for improving the lifetime of an object tracking sensor network. One is to reduce energy consumption in the sensing component (e.g., [18], [35]). 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 [31], [33], the sensor nodes are organized into a cluster-based architecture such that a cluster head calculates object locations based on signal readings from its slave nodes. Based on these studies, we assume object locations can be obtained by cluster heads and only focus our 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. Nevertheless, the focuses of the prior studies are different from ours. Goel and Imielinski [9] proposed a prediction-based approach. 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 one. Xu *et al.* [32] suggested a dual-prediction scheme where a fixed prediction model is deployed at both the base station and the sensor nodes. These studies complement our work in that prediction can be incorporated into our EASE scheme to further reduce remote update traffic. Kung and Vlah [17] investigated continuous location queries and proposed a publish-and-subscribe tracking method. In contrast, we leverage error tolerances to improve network performance for one-shot location queries. While there has been research on the trade-off between energy conservation and tracking quality (e.g., [10], [24]), the trade-off has not been investigated in the dissemination of location data, which is the topic of this paper.

Our work also bears some similarity to location management for mobile networks [4]. As the purpose of location management is to locate roaming users for call delivery, locations are managed at a fixed granularity (i.e., cell). In contrast, our architecture is capable of adaptively storing location data at different degrees of accuracy to improve the efficiency of query processing.

B. Data Storage and Query Processing

A simple storage model is to have a centralized base station collect and store the sensed data. This approach is good for aggregate data collection (e.g., sum, average, maximum, and median) [21], [28], where excessive sensed data can be pruned during aggregation along the routing path, and only short summaries are maintained at the base station. However, this is not efficient for non-aggregate data collection (e.g., the location queries considered in this paper). The base station and the sensor nodes around it can easily become hotspots, which would shorten the network lifetime.

Recently, in-network storage has been advocated in many research projects. In the TinyDB project, Madden *et al.* [20] presented pull-based acquisitional query processing (ACQP), where the sensor nodes control where, when, and how often data are acquired and delivered to query operators. The Cougar project [7] employed a hybrid pull-push model, in which sensed data are pushed to some selected view nodes, from where the data are pulled to answer queries. Ratnasamy *et al.* [26] proposed an in-network data-centric storage model: sensed data are pushed to the sensor node nearest to some geographical location hashed from a predefined key. Zhang *et al.* [36] suggested storing sensed data locally. A centric ring-based index was proposed to facilitate query processing. More recently, Lu *et al.* [19] proposed a spatiotemporal query service called MobiQuery to allow mobile users to query their surrounding areas through a sensor network. Jiang and Jin [14] developed robust aggregation techniques for extracting statistical information from sensor networks. Unfortunately, none of these prior studies have examined the ability of approximate data storage to improve energy efficiency.

C. Approximate Query Processing

Early work on approximate query processing focused on a wired network [23]. Han *et al.* [11] conducted a pioneering study on answering 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 [11] are not applicable to object tracking applications. This is because [11] considered only a simple single-hop system where each sensor node communicates with the server directly and each target phenomenon is always captured by a fixed sensor. In contrast, we consider a dynamic sensing scenario where the location of a moving object is acquired by different nodes at different times, thus a cooperative location updating protocol is needed. Moreover, we consider a multi-hop sensor network. As a result, the cost of location updates/queries also varies according to the locations of moving objects. These differences make our system modeling and performance analysis completely different from those in [11].

Precision-constrained queries [6], [27], [30] have also been studied for in-network data aggregation, which has a different focus from object tracking sensor networks. Compressing historical sensor readings for transmission also saves energy [5]. However, it is applicable to querying historical data only. In contrast, we consider applications that are interested in querying the current locations of moving objects.

D. Data Routing

A sensor network is typically connected by wireless links in an ad hoc manner. To relay data in a sensor network, many routing algorithms have been proposed to address energy efficiency, scalability, and reliability issues. They can be classified into three categories: *data-centric*, *hierarchical*, and *location-based routings* [3]. In data-centric routing (e.g., directed diffusion [13]), the sink floods the query to a certain region of interest and the sources report data to the sink through the route established based on the named query. Due to the high cost of flooding, such a protocol is suitable for long-lived queries only. Hierarchical protocols (e.g., LEACH [12]) group the sensor nodes into clusters such that a cluster head performs data aggregation/fusion and communicates with other heads on behalf of the nodes within its cluster. Location-based protocols (e.g., GPSR [15]) make use of geographical position information to transport data.

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; for example, a cluster head can triangulate object locations based on signal readings from its slave nodes [31], [29], [33]. The object locations are sampled at a fixed sampling rate. We also assume that each moving object being tracked has a unique identifier. Since this paper aims at energy-efficiently storing and disseminating object location data in support of approximate location queries, we shall focus on reducing communication among cluster heads. Unless explicitly specified, a sensor node refers to a cluster head in 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 the *querying node*) from anywhere in the network. Each approximate location query is specified by a tuple $<$*object_id*, $p>$, where *object_id* is the identifier of the target object, and p is the error in object location that the query can tolerate.[2]

Local and Centralized Storage. Intuitively, the object location data can be stored at 1) the sensor node that detected the object; or 2) a centralized storage node, which could be either a centralized base station or a sensor node determined by the data-centric storage scheme (DCS) [26] (see Figure 1(a)).[3] If the location data are stored locally at the detecting node (known as local storage (LS for short)), a query that wants to find the location of some object has to be *flooded* over the whole network. Thus, the query cost is high. In contrast, if the centralized storage (CS for short) scheme is adopted in the system, any location update of a moving object should be sent to the centric storage node. This results in high update traffic. In the next section, we propose a hybrid in-network storage scheme that achieves a good balance between the query cost and update cost for approximate location queries. Without loss of generality, we shall assume for the rest of this paper that the DCS scheme [26] is employed for centralized storage.

IV. EASE: Energy-Conserving Approximate Storage

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

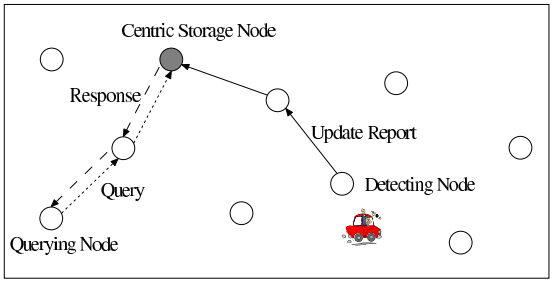
A. Overview

The EASE scheme attempts to cut down the update traffic by maintaining two versions of the location data for each object: an accurate version at a local storage node and an approximate version at the centric storage node. The approximate location of an object at the centric storage node is bounded by an *approximation radius*. A stored location o with an approximation radius r means that the object must be in an *approximation area* defined by a circle of radius r centering at o.

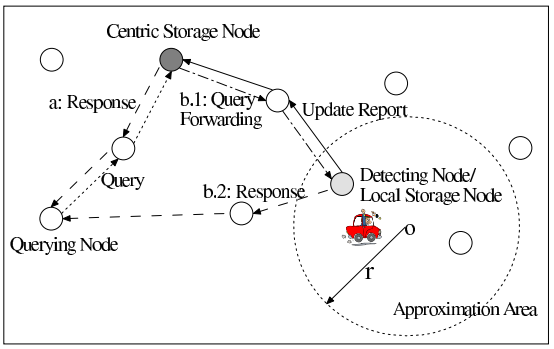
In addition to storing the approximation location of an object, the centric storage node also keeps track of the local

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

[3]In DCS, the centric storage node of an object is determined by applying a pre-defined hash function to the object identifier.



(a) Centralized Storage (CS)



(b) Energy-Conserving Approximate Storage (EASE)

Fig. 1. Illustration of Query Processing Procedure

storage node of the object. That is, the centric storage node maintains two attributes for an object: the approximate location and the local storage node. The former is used to answer queries with less stringent precision constraints while the latter is used to forward queries with more stringent precision constraints (without this information, the local storage nodes can only be found by flooding). Upon receiving an approximate location query with an error tolerance p, the centric storage node compares it against the 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

Initially, the local storage node of an object is the node that first detected it. Every sensor node in the approximation area is notified of the local storage node. At each subsequent sampling instance, if the object location detected is within its current approximation area, the detecting node sends a *local update* including the new object location to the local storage node as shown in Figure 2(a). Otherwise, if the object moves out of the current approximation area, the detecting node elects itself as the *new local storage node* and sends a *remote update* including its own id/location and the new object location to the centric storage node (see Figures 2(b) and 2(c)). It is likely that the object is still in the current approximation area, but the detecting node is not aware of the current local storage node (e.g., the detecting node is outside the approximation area).

Algorithm 1 Protocol Executed at Centric Storage Node.

```
1:  if receiving a query <object_id, p> then
2:      if p ≥ r then
3:          return the stored location to the querying node;
4:      else
5:          forward the query to the local storage node of the
            queried object;
6:      end if
7:  end if
8:  if receiving a remote location update message then
9:      if the old approximation area is unknown at the local
        storage node then
10:         send an invalidation geocast message to the old
            approximation area;
11:     end if
12:     store the new location and local storage node of the
        object;
13: end if
```

Algorithm 2 Protocol Executed at Local Storage Node.

```
1: if receiving a forwarded query <object_id, p> then
2:     return the stored location to the querying node;
3: end if
4: if receiving a local location update message then
5:     store the new location of the object;
6: end if
```

In this case, the detecting node also elects itself as the new local storage node and sends a remote update to the centric storage node. A new approximation area is then formed as a circle centered at the new object location.

To notify the sensor nodes in the new approximation area, the new local storage node sends out a *notification* geocast [22] message including its own id/location and the new approximation area (see below for how it works). If the new local storage node is aware of the previous local storage node (and hence the obsolete approximation area), it also sends an *invalidation* 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 new local storage node is not aware of the previous local storage node, it informs the 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)). Given an approximation radius r, Algorithms 1 through 3 summarize the protocols executed at the centric storage node, the local storage node, and the detecting node, respectively.

The purpose of geocast is to send a message to all the nodes in a given geographical area. It works in two phases. In the first phase, the message is routed towards the target area using some geographical routing protocol such as GPSR [15]. If the geocast initiator is within the target area, the first phase is not needed. After reaching the target area, the message is flooded to all sensor nodes in the area through broadcast. On receiving the message, a sensor node further broadcasts it to the neighbors only if the message has not been received before

Algorithm 3 Protocol Executed at a Node Detecting an Object Located at o.

1: **if** o is in the valid approximation area **and** the local storage node is known **then**
2: send a location update to the local storage node;
3: **else**
4: send a location update to the centric storage node; piggyback the information whether the old approximation area is known or not;
5: elect itself as a new local storage node;
6: set the new approximation area $A' \leftarrow$ a circle centered at o with radius r;
7: **if** the old approximation area A is known **then**
8: send an *invalidation* geocast message to A;
9: **end if**
10: send a *notification* geocast message to A';
11: **end if**

and the sensor node is within the target area. Recall that in EASE, when the local storage node changes, a notification message and an invalidation message are geocast to the new and obsolete approximation areas, respectively. In order to avoid redundant message broadcast in the overlap region of these two areas, we do not distinguish the notification message from the invalidation message in implementation. A geocast message specifying both the obsolete and new approximation areas are sent to these two areas at the same time. On receiving the geocast message, if the node is in the new approximation area, it records the new local storage node. Otherwise if the node is in the obsolete approximation area, it removes the recorded old local storage node.

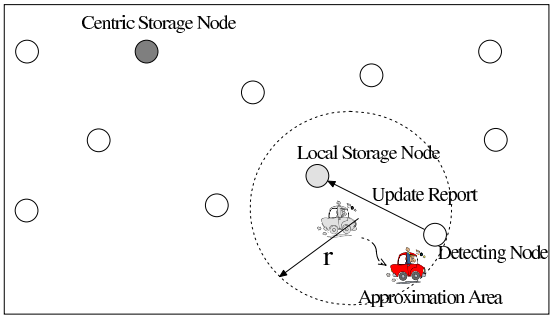
C. Discussion

Wireless sensor networks are unreliable in nature. In this section, we discuss the strategies that EASE can employ to deal with node failures and message losses. Note that these issues are faced by any data dissemination schemes including LS and CS.

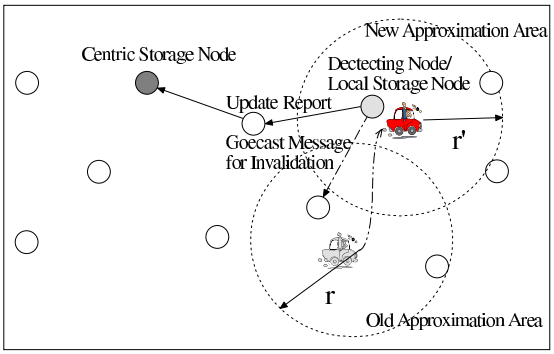
The EASE scheme relies on a centralized base station or data-centric storage (DCS) to store approximate location data. To tolerate node failures, DCS enhances the centric storage nodes by replication [8], [26]. EASE stores accurate location data at local storage nodes, which can also be replicated to improve reliability and availability using similar techniques employed by DCS.

As the link loss rate is high in sensor networks, hop-by-hop recovery (e.g., by link-level retransmission) has been suggested to remedy message losses [16]. In addition, message losses can be handled at the application layer. For example, for query messages, we set a TTL value (i.e., twice the round-trip time between two farthest nodes in the network). If a query is not answered within the TTL time, the application may assume the previous query has got lost in the network and resend the query.

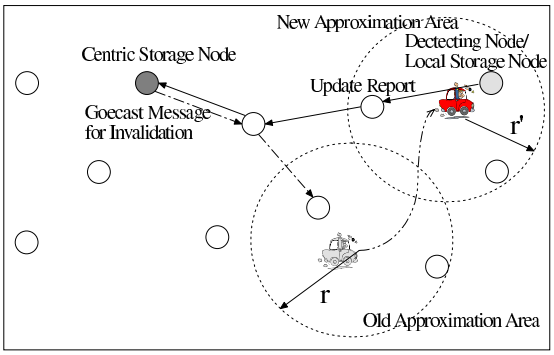
There are two types of geocast messages in EASE (i.e., invalidation and notification). If a notification message is lost,



(a) Local Update



(b) Remote Update: Case I



(c) Remote Update: Case II

Fig. 2. Location Update Dissemination with EASE

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 elect itself as a new local storage node and send a remote update to the centric storage node. If an invalidation message is lost, some sensor nodes may not be aware of the change of local storage node. We can request the previous local storage node to acknowledge the receipt of the invalidation message. The new local storage node will retransmit the message until it is acknowledged. For other nodes in the obsolete approximation area, if they do not receive the invalidation message and continue to send the local update to the previous local storage node, the previous local storage node will notify them of 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 (Section V-A). Then, we study the setting of the approximate radius for EASE. When the

Notation	Description
n	number of sensor nodes
λ	query rate
μ	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_{max}	maximum level of 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

mobility pattern is available, we analyze the optimal setting of the approximation radius (Section V-B). When the mobility pattern is not available, we devise an adaptive algorithm to dynamically adjust the approximation radius (Section V-C).

A. Performance Analysis

This section analyzes the performance of EASE in terms of message complexity (i.e., the total number of message transfers 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 λ and the sensor sampling rate is μ. Let C_{qn} and C_{qf} be the costs of answering a query by the centric storage node and the local storage node, 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

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

where $p_{qf}(r)$ is the probability that the approximate location stored at the centric storage node does not satisfy the query-specified precision requirement and $\eta(r)$ is the rate of remote updates (i.e., the rate the object moves 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 [26], [36], we use n to approximate the cost of flooding a message over the whole network, and $\sqrt{n}$ to approximate the cost of sending a message between two nodes in the network. It is easy to see (from Figure 1(b)) that we obtain

$$C_{qn} = 2\sqrt{n}, \tag{2}$$

$$C_{qf} = 3\sqrt{n}. \tag{3}$$

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 the local storage node sending the update to the centric storage node and two geocast messages to notify (invalidate) the new (obsolete) approximation area. The remote update cost is approximated by $\sqrt{n}$. We approximate the geocast cost by the number of sensor nodes in the target area; that is, $\pi r^2 f$, where r is the approximation radius and f is the density of sensor nodes. Therefore, we obtain C_{ur} as

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

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

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

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

$$C_{ul} = \frac{2}{3}r\sqrt{f}. \tag{6}$$

We assume that the error tolerances of the queries are uniformly distributed in the range of [0, p_{max}]. Thus, the probability of a query not being satisfied by the centric storage node is

$$p_{qf}(r) = min\{\frac{r}{p_{max}}, 1\}. \tag{7}$$

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

$$\begin{aligned} C(r) &= \lambda\cdot 2\sqrt{n} + min\{\frac{r}{p_{max}}, 1\}\cdot\lambda\cdot\sqrt{n} \\ &\quad +\eta(r)\cdot(\sqrt{n}+2\pi r^2 f) + \mu\cdot\frac{2}{3}r\sqrt{f}. \end{aligned} \tag{8}$$

As can be seen, the overall message complexity is basically a function of approximation radius r. The next two subsections study the setting of r in detail.

B. Optimal Approximation Setting with Known Mobility Pattern

This section studies the optimal approximation setting based on the mobility pattern. We assume a 2-dimensional random walk model [17], in which the object moves in steps.[4] At each step, the object moves a distance of d along an arbitrary direction (i.e., with angle θ uniformly distributed in [0, 2π]). 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 for details):

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

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

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

[4]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.

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

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

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

$$r^* = \begin{cases} \sqrt[3]{\frac{6p_{max}d^2\sqrt{n}}{l\cdot(3\lambda\sqrt{n}+2\mu p_{max}\sqrt{f})}} & \text{if } r \le p_{max}, \\ \sqrt[3]{\frac{3d^2\sqrt{n}}{l\mu\sqrt{f}}} & \text{if } r > p_{max}. \end{cases} \quad (12)$$

The one producing the lower message complexity $C(r)$ will be selected as the final setting of r^*. From (12), we can observe that the optimal setting of r^* is affected by many factors such as the network size, query rate, sensor sampling rate, mobility pattern, and precision requirement. Intuitively, a faster movement (i.e., larger d or smaller l) results in a larger r^* so as to reduce the remote update traffic; on the other hand, a smaller r^* is desired at a higher sensor sampling rate (i.e., higher μ) in order to reduce the local update traffic. Also note that if r^* is set greater than p_{max}, all queries will be relayed to the local storage node via the centric storage node. In this case, the approximate data storage is used to maintain the location of the local storage node only.

C. Adaptive Approximation Setting with Unknown Mobility Pattern

It is obvious that the larger the approximation radius r, the lower the rate of moving out of the approximation area $\eta(r)$. Following [11] and [23], we assume that $\eta(r)$ is proportional to $1/r^2$; that is, $\eta(r) = \frac{K}{r^2}$, where K is a parameter depending on the mobility model. We assume that p_{max} is sufficiently large, in which case $p_{qf}(r) = \frac{r}{p_{max}}$. Using the optimization technique described in Section V-B, we can derive the optimal r^* for $r \le p_{max}$:

$$r^* = \sqrt[3]{\frac{6p_{max}K\sqrt{n}}{3\lambda\sqrt{n} + 2\mu p_{max}\sqrt{f}}}. \quad (13)$$

It is not difficult to observe that when $r = r^*$,

$$\frac{p_{qf}(r)}{\eta(r)} = \frac{6\sqrt{n}}{3\lambda\sqrt{n} + 2\mu p_{max}\sqrt{f}} \quad (14)$$

is a constant (denoted by ρ). Motivated by this observation, we propose an adaptive algorithm to dynamically adapt approximation radius r. The basic idea is to maintain the observed ratio of $p_{qf}(r)$ to $\eta(r)$ at ρ. To do so, the centric storage node keeps track of the query forwarding probability $p_{qf}(r)$. It also maintains the remote update rate using an exponential aging method. At each remote update, the estimate of the remote update rate is adjusted as

$$\eta(r)^{new} = (1-\alpha)\cdot\eta(r)^{old} + \alpha \cdot \frac{1}{T_c - T_l}, \quad (15)$$

where T_c is the current time, T_l is the time of the last remote update, and α is a factor weighing the importance of the current update against past updates. The setting of r is then adjusted by the centric storage node based on Algorithm 4. Recall that $p_{qf}(r)$ is proportional to r, and $\eta(r)$ is inversely proportional to r^2. If the observed ratio is greater than $\rho\cdot(1+\epsilon)$, we decrease r by a factor of (1+δ); if the observed ratio is lower than $\rho\cdot(1-\epsilon)$, we increase r by a factor of (1+δ). On computing the new approximation radius, the centric storage node informs the new local storage node of the radius. On receiving the radius, the new local storage node performs the notification geocast (see Section IV-B). We shall evaluate the performance of this adaptive setting algorithm in the next section.

Algorithm 4 Adjustment of Approximation Radius (at Centric Storage Node).

```
for each remote update do
    update the rate of remote updates η(r) according to (15);
    if p_qf(r)/η(r) > ρ·(1+ε) then
        set r' ← r/(1+δ);
    else if p_qf(r)/η(r) < ρ·(1−ε) then
        set r' ← r·(1+δ);
    end if
    return r' to the new local storage node;
end for
```

VI. Performance Evaluation

A. Simulation Setup

In this section, we conduct simulation experiments to compare the proposed EASE scheme with the conventional storage schemes. We have developed a simulator based on ns-2 [1] and NRL's sensor network extension [2]. Table II summarizes the system parameters and their settings used in our experiments.

We simulate 225 sensor nodes as deployed in a 500×500-m^2 field. The field is divided into 225 34×34-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 [31], [33]. Like in the previous work [13], [26], the radio transmission range is set at 40 m. In order to save the energy spent in idle listening, the simulator employs the B-MAC [25] in its MAC layer. Specifically, a sensor node can send a message at any time by including a preamble. Each sensor node periodically wakes up (in idle listening mode) to sample the channel. If some incoming message is detected, it stays awake to receive the message. Otherwise, the node enters the sleeping mode. The simulator also includes an energy model that measures the energy consumption of each sensor node. Approximate location queries are issued randomly from the sensor nodes in the field. The query rate is set at 1 / sec by default. The error tolerances of the queries are randomly distributed between 0 and p_{max}. The default setting of p_{max} is 50 m.

To facilitate geographical message routing (such as for update reporting and query forwarding), the greedy perimeter stateless routing (GPSR) protocol [15] is employed by the simulator. GPSR operates in two modes: greedy mode and perimeter mode. In the greedy mode, a sensor node forwards a

Parameter	Setting
Field Size	500×500 m^2
Number of Nodes (n)	225
Node Density (f)	1 node / 34×34 m^2
Radio Range	40 m
Power Consumption in Sending Messages	60.0 mW
Power Consumption in Receiving Messages	45.0 mW
Power Consumption in Idle Listening	15.5 mW
Power Consumption in Sleeping	0.09 mW
B-MAC Preamble Length	271 $bytes$
B-MAC Wake-up Interval	100 $msec$
B-MAC Sampling Time	2.55 $msec$
Data Transfer Rate	0.416 $msec/byte$
GPSR Beacon Interval	3 sec
GPSR Beacon Expiration	13.5 sec
GPSR Implicit Beacon	yes
Sensor Sampling Rate (μ)	5 /sec
Query Rate (λ)	0.1 - 10 /sec
Maximum Level of Error Tolerance (p_{max})	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	Random / Semi-Random Walk			Random Waypoint	
	d	Init Speed	l	Speed Range	Pause
Slow	1 m	1 m/sec	1 sec	1 - 2 m/sec	1 sec
Moderate	5 m	5 m/sec	1 sec	5 - 8 m/sec	1 sec
Fast	15 m	15 m/sec	1 sec	10 - 20 m/sec	1 sec
Mixed	-	-	-	1 - 20 m/sec	1 sec

TABLE III
MOBILITY MODELS AND PARAMETER SETTINGS

message to a neighbor closer to the destination than itself. If no such neighbor exists, the algorithm switches to the perimeter mode, which recovers by routing around the perimeter of the region.

Three mobility models, *random walk*, *semi-random walk*, and *random waypoint*, are used to model the moving pattern of objects in the simulation. The random walk model has been described in Section V-A. The semi-random walk is similar to the random walk. However, unlike the random walk, after each movement step, the direction is adjusted by a random small angle ϵ_θ (uniformly distributed between $[-\frac{\pi}{8}, \frac{\pi}{8}]$), and the speed is adjusted by a random small percentage ϵ_v (uniformly distributed between $[-5\%, 5\%]$). In the random waypoint model, an object selects a destination at random in the simulated field, and moves to the destination at a speed randomly chosen from a configured range; upon arrival, it pauses for a random period and selects a new destination. The random walk model is suitable for simulating small-scale scenarios, while the random waypoint model is more suitable for large-scale on-purpose movements [17]. The mobility profiles used in the experiments, including *slow*, *moderate*, *fast*, and *mixed*, are listed in Table III.

We compare EASE with Centralized Storage (CS) and Local Storage (LS) (described in Section III). The performance is measured in terms of *message complexity* (# messages transferred in the network) and *energy consumption* of the sensor nodes. Similar to [26], we do not measure the message overhead incurred by the underlying routing protocol (e.g., beacons in GPSR). Such an overhead is usually of lower order than the application data traffic. To facilitate an easy illustration and performance comparison, we report *normalized cost* for each performance metric. The normalized cost of a scheme is defined as the ratio of the measured cost of the scheme to that of CS. The smaller is the normalized cost, the better does the scheme perform. Each simulation run lasted for 500 seconds of simulated time; the first 20 seconds were considered the warm-up period to eliminate initialization effects such as speed decay [34].

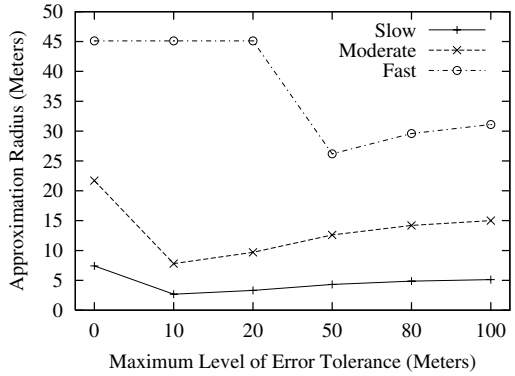
B. Optimal Setting of the Approximation Radius

The optimal approximation setting of EASE, r, can be obtained using the optimization technique described in Section V-B under the assumption of random walk mobility. Figure 3(a) shows the optimal settings of r for slow, moderate, and fast random walk mobility profiles under a number of increasing error tolerances. When the error tolerance is low (i.e., $p_{max} = 0$ for *slow* and *moderate* profiles; $p_{max} \leq 20$ for *fast* profile), EASE selects a large r ($> p_{max}$) 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 error tolerance becomes higher, EASE chooses a smaller r ($< p_{max}$) to limit query forwarding, thus reducing the overall traffic. When the error tolerance is further increased, approximation radius r is slightly increased to reduce the update traffic.

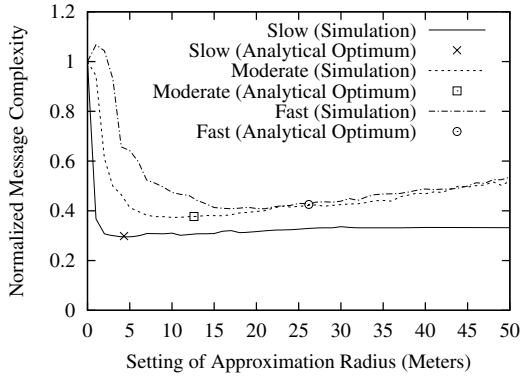
Next, we compare the above approximation settings obtained from the theoretical analysis with those obtained from simulation experiments. In the simulation, we vary the approximation radius from $0, 1, 2, \cdots$, till 50 and test the performance of each radius. The results for $p_{max} = 50$ are plotted in Figure 3(b). The analytical approximation radii are also shown in the figure. As can be seen, the analytical approximation settings are close to the actual optimal points and approach the optimal performances. This validates our analysis of the optimal approximation setting.

C. Message Complexity

This section examines the message complexity of EASE with the optimal approximation setting. We use the random walk mobility model in this set of experiments. Figure 4(a) shows the normalized message complexity under various error tolerances. Neither LS or CS are aware of the system workload. LS has the highest message complexity, that is, 6.36, which is not plotted in the figure. In contrast, EASE is able to set a proper approximation radius in optimizing the network traffic based on the query and update patterns. As a result, EASE improves the performance over CS by 49%-72%. Even when the error tolerance is set at zero, by selecting a large approximation radius, as shown in Figure 3(a), EASE significantly reduces the update traffic at the cost of slightly

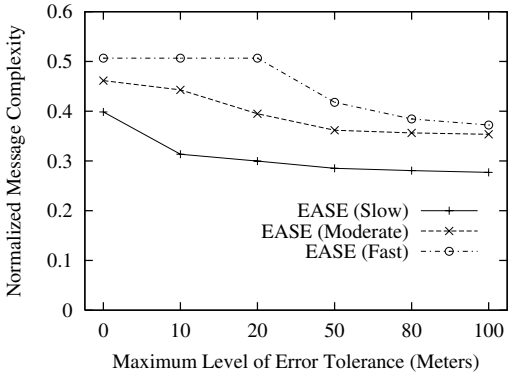


(a) Approximation Setting Obtained from Theoretical Analysis

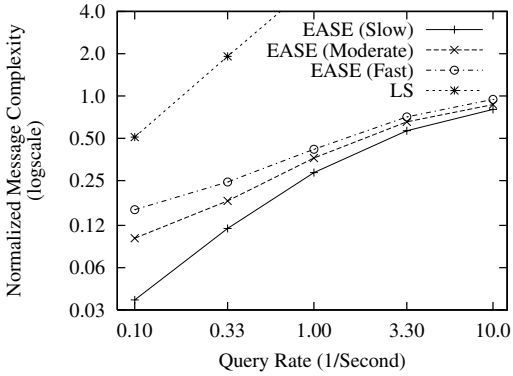


(b) Simulation Results ($p_{max} = 50m$)

Fig. 3. Optimal Setting of Approximation Radius ($\lambda = 1\ /\ sec$)



(a) Message Complexity vs. Query Precision Requirement ($\lambda = 1\ /\ sec$)



(b) Message Complexity vs. Query Rate ($p_{max} = 50m$)

Fig. 4. Message Complexity

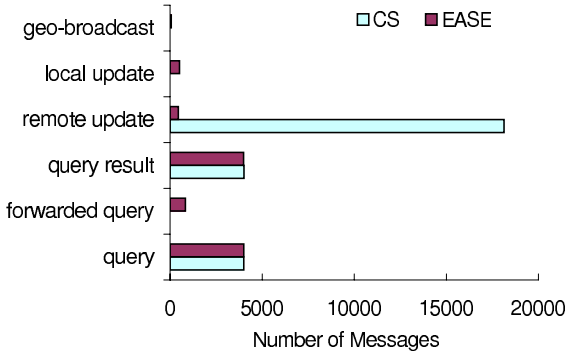


Fig. 5. Breakdown of Message Complexity (*Moderate*, $p_{max} = 50m$)

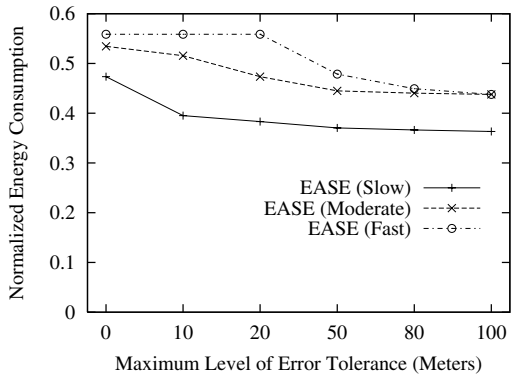
increased query traffic, and achieves an improvement of 49%-60% in the overall traffic. In general, the higher the error tolerance, the greater is the improvement achieved by EASE. This is expected since EASE is designed to exploit the error tolerance in location queries. A higher error tolerance implies 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 location changes result in local updates which are propagated only to the local storage nodes.

To gain more insight into how EASE improves the performance over CS, we provide a breakdown of the traffic in Figure 5, where p_{max} is set at 50 m. In the CS scheme, a significant amount of traffic contributes to remote updates. By keeping imprecise data at the centric storage node, EASE reduces remote updates by several orders of magnitude. Figure 5 also shows that EASE's overhead for forwarding queries and geocasting is trivial compared to the reduced update traffic. Overall, EASE reduces the total message complexity for CS by 62%.

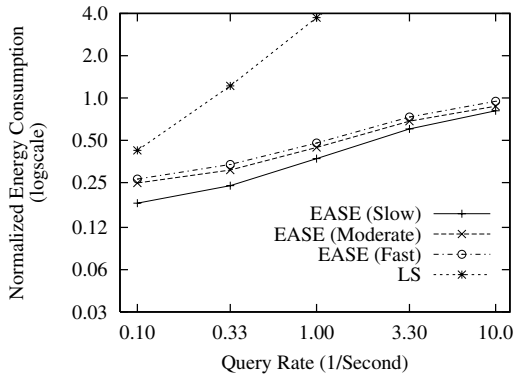
Figure 4(b) shows the normalized message complexity as a function of the query rate when p_{max} is fixed at 50 m. LS performs the worst. For CS, when the query rate is low, most network traffic is due to location updates. EASE significantly cuts down the update traffic by setting a large approximation radius and reduces the overall message complexity by up to 96%. When the query rate is extremely high (e.g., 10), the approximation radius is set close to zero, and hence EASE has a performance similar to CS.

D. Energy Consumption

We now proceed to evaluate the energy consumption of EASE with the optimal approximation setting. Figures 6(a) and 6(b) show the total energy consumed by the sensor network during a simulation run, which is normalized by that of CS. In Figure 6(a), LS has the highest normalized energy (i.e., 3.73) and is not plotted. Basically, the trends for energy consumption are similar to those for message

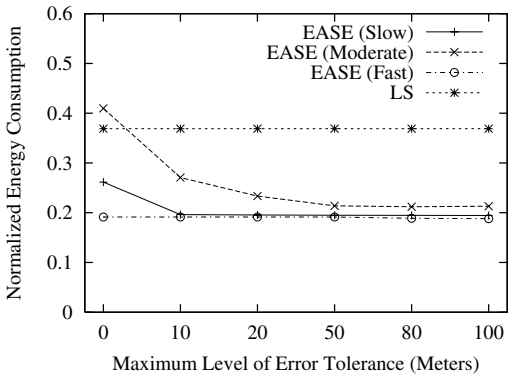


(a) Energy Consumption vs. Query Precision Requirement ($\lambda = 1 \ / \ sec$)

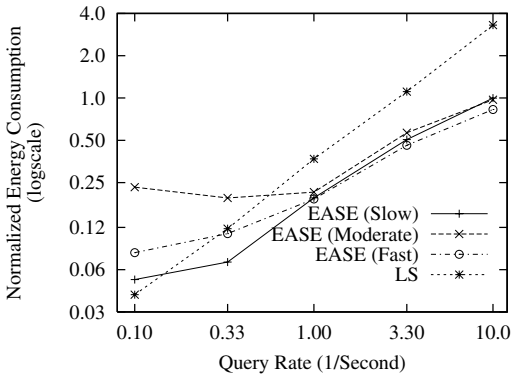


(b) Energy Consumption vs. Query Rate ($p_{max} = 50m$)

Fig. 6. Total Energy Consumption



(a) Highest Energy Consumption vs. Query Precision Requirement ($\lambda = 1 \ / \ sec$)



(b) Highest Energy Consumption vs. Query Rate ($p_{max} = 50m$)

Fig. 7. Energy Consumption of the Most-Consuming Node

complexity (Figures 4(a) and 4(b)). However, the performance improvement of EASE over LS and CS is less in terms of energy consumption. This is partly because in addition to sending and receiving messages, the sensor nodes spend energy in idle listening and sleeping. The latter portion of the energy cost is similar for all schemes.

We also measure the energy consumption of 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 for Figure 7. We can see that EASE performs the best in all cases except when the query rate is very low (leftmost points of Figure 7(b)). In most cases, the energy consumption of EASE is only $20\%-50\%$ of that of CS. This implies that EASE can prolong the network lifetime over CS by a factor of $2-5$. It is also 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 node is a hotspot. However, with reduced update traffic by EASE, the hotspot is no longer the centric storage node. Instead, some local storage nodes become hotspots. On one hand, a faster moving object generates more update traffic. On the other hand, a faster moving object changes local storage nodes more frequently and hence balances the loads of the sensor nodes. Combining these effects, EASE has a longer network lifetime for *slow* and *fast* objects than for *moderate* objects.

E. Query Latency

The CS scheme is expected to have a very good performance in query latency because the accurate location data are always available at the centric storage node, 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 latency of answering approximate location queries to examine how well EASE can perform in query latency.

As shown in Figure 8, when the error tolerance is low (i.e., $p_{max} = 0$, the worst case), EASE performs worse than CS by no more than 52%. However, as the error tolerance increases, EASE consistently improves the query latency (and even outperforms CS for large values of p_{max}). This is mainly because for a larger value of p_{max}, EASE incurs less network traffic (as shown in Figure 4(a)) and hence less message transmission collisions, which reduces the overall latency even though the querying path is lengthened.

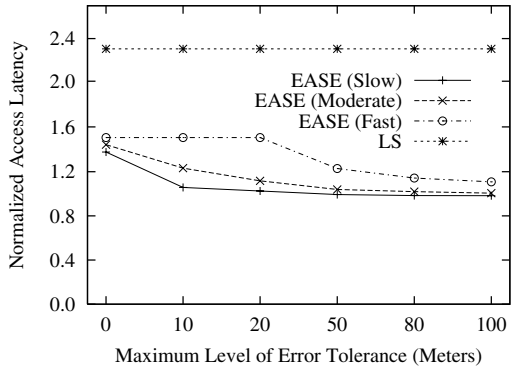
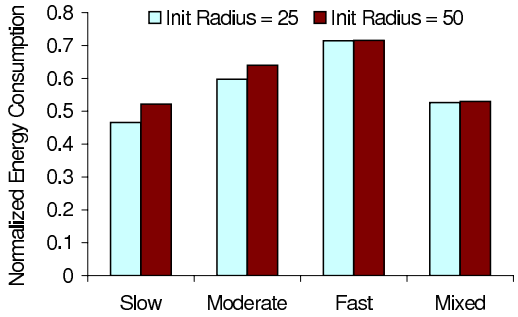


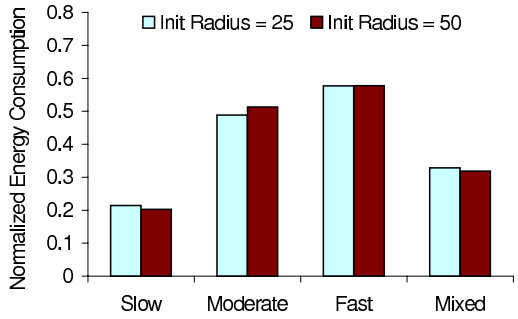
Fig. 8. Query Latency vs. Query Precision Requirement ($\lambda = 1$ / sec)

F. Performance of Adaptive Setting Algorithm

Finally, we evaluate the performance of the adaptive algorithm for setting approximation radius r (Section V-C). Radius r is initialized by some seed value, and is updated by the proposed adaptive algorithm afterwards. We set α, ϵ, and δ at 0.5, 0.1, and 0.1 respectively in the adaptive algorithm. Figure 9 shows the normalized energy consumption for the four mobility profiles of random waypoint mobility when the maximum error tolerance p_{max} is set at 50 m and the query rate λ is set at 1 /sec. We test two methods to seed r; that is, initializing r with the average error tolerance (i.e., 25) or the maximum error tolerance (i.e., 50). As can be seen, the adaptive algorithm works effectively in both cases tested. EASE improves over CS by 30%-50% in terms of total energy consumption, and its most-consuming node spends only 20%-60% of the energy of CS's (i.e., extends the lifetime of the sensor network by a factor of 1.5−5). The two initial values of r achieve a similar performance. Figure 10 shows the message complexities, measured every 10 seconds, of CS and EASE (with two different initial values) as a function of simulated time. It is clear that after the warm-up period (i.e., after 100 seconds), the two EASE schemes converge. This also implies that the adaptive algorithm is not sensitive to the initial setting of r. Similar performance trends can be observed for the semi-random mobility model. As shown in Figure 11, EASE substantially outperforms CS in terms of energy consumption.



(a) Total Energy Consumption



(b) Energy Consumption at Most-Consuming Node

Fig. 9. Performance of Adaptive Settings (Waypoint Mobility)

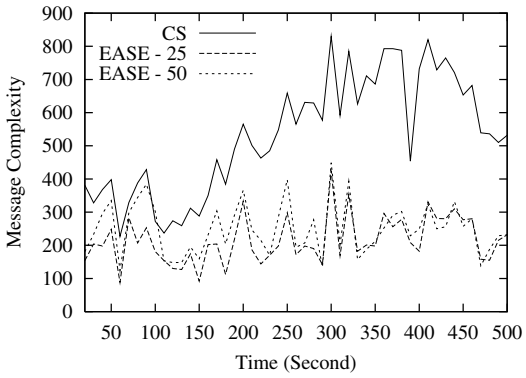


Fig. 10. Message Complexity along Time Series (Waypoint, *Slow* Mobility Profile)

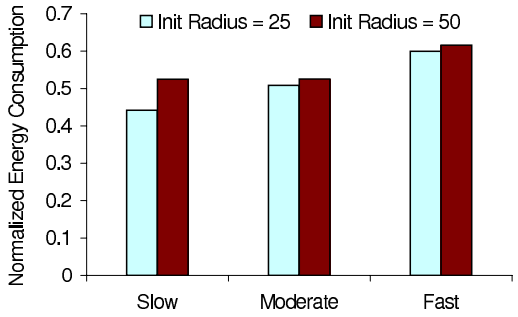
VII. Conclusions

This paper presents a study on the processing of precision-constrained approximate location queries for object tracking sensor networks. An energy-efficient 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 the approximation radius when the object mobility pattern is known, and devised an adaptive algorithm to adjust the approximation setting when the mobility pattern is not available or is dynamically changing. We have also evaluated the proposed techniques through extensive simulation-based experiments. The experimental results validated our theoretical analysis of the optimal approximation setting. They also demonstrated that the EASE scheme, working together with the proposed approximation setting methods, saves significant energy for sensor networks and prolongs the network lifetime.

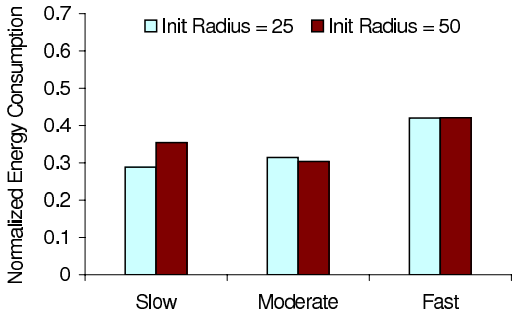
With regard to future work, we plan to extend EASE to answering approximate spatial queries such as finding k nearest neighbors. Besides one-shot queries, we will also investigate continuous monitoring queries. In addition, we are planning to develop a testbed using Berkeley Motes to validate the proposed schemes.

Acknowledgments

This work was supported by the Research Grants Council of the Hong Kong SAR, China under Project Nos. HKBU211505, HKBU211206, HKBU211307, and FRG/05-06/II-65. Xueyan Tang's work was supported in part by Nanyang Technological University under Project No. RG47/06. Wang-Chien Lee was



(a) Total Energy Consumption



(b) Energy Consumption at Most Consuming Node

Fig. 11. Performance of Adaptive Settings (Semi-Random Mobility)

supported in part by the National Science Foundation under Grant no. IIS-0328881, IIS-0534343, and CNS-0626709

REFERENCES

[1] The network simulator - ns-2. [Online]. Available at http://www.isi.edu/nsnam/ns/.

[2] NRL's sensor network extension to ns-2. [Online]. Available at http://nrlsensorsim.pf.itd.nrl.navy.mil/.

[3] K. Akkaya and M. Younis. A survey of routing protocols in wireless sensor networks. *Elsevier Journal of Ad-Hoc Networks*, 2004.

[4] I. F. Akyildiz, J. McNair, J. Ho, H. Uzunalioglu, and W. Wang. Mobility management for next generation wireless systems. *Proceedings of the IEEE*, 87(8):1347–1384, August 1999.

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

[6] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Processing approximate aggregate queries in wireless sensor networks. *Information Systems*, 31(8):770–792, December 2006.

[7] A. Demers, J. Gehrke, R. Rajaraman, J. Trigoni, and Y. Yao. The Cougar project: A work-in-progress report. *ACM SIGMOD Record*, 32(4), Dec. 2003.

[8] 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.

[9] S. Goel and T. Imielinski. Prediction-based monitoring in sensor networks: Taking lessons from MPEG. *ACM Computer Communication Review*, 31(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] Q. Han, S. Mehrotra, and N. Venkatasubramanian. Energy efficient data collection in distributed sensor environments. In *IEEE ICDCS*, Tokyo, Japan, March 2004.

[12] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless sensor networks. In *Conference on System Sciences*, Hawaii, Jan. 2000.

[13] 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.

[14] H. Jiang and S. Jin. Scalable and robust aggregation techniques for extracting statistical information in sensor networks. In *IEEE ICDCS*, Lisboa, Portugal, July 2006.

[15] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless sensor networks. In *ACM MobiCom*, Boston, MA, August 2000.

[16] S. Kim, R. Fonseca, and D. Culler. Reliable transfer on wireless sensor networks. In *IEEE SECON*, October 2004.

[17] H. T. Kung and D. Vlah. Efficient location tracking using sensor networks. In *IEEE WCNC*, New Orleans, LA, March 2003.

[18] H. Liu, X. Jia, P. Wan, C.-W. Yi, S. Makki, and P. Niki. Maximizing lifetime of sensor surveillance systems. *IEEE/ACM Transactions on Networking (ToN), to appear*, 2006.

[19] C. Lu, G. Xing, O. Chipara, C.-L. Fok, and S. Bhattacharya. A spatiotemporal query service for mobile users in sensor networks. In *IEEE ICDCS*, Columbus, OH, June 2005.

[20] 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 (TODS)*, 30(1):122–173, 2005.

[21] 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.

[22] J. C. Navas and T. Imielinski. GeoCast – geographic addressing and routing. In *ACM/IEEE MobiCom*, 1997.

[23] C. Olston, B. T. Loo, and J. Widom. Adaptive precision setting for cached approximate values. In *ACM SIGMOD*, Santa Barbara, CA, May 2001.

[24] S. Pattem, 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.

[25] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *ACM MobiSys*, Baltimore, MD, November 2004.

[26] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with GHT, a geographic hash table. *ACM/Kluwer MONET*, 8(4), 2003.

[27] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. Balancing energy efficiency and quality of aggregate data in sensor networks. *VLDB Journal*, 13(4), December 2004.

[28] N. Shrivastava, C. Buragohain, and D. Agrawal. Medians and beyond: New aggregation techniques for sensor networks. In *ACM SenSys*, Baltimore, MD, Nov. 2004.

[29] D. Smith and S. Singh. Approaches to multisensor data fusion in target tracking: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 18(12):1696–1710, December 2006.

[30] X. Tang and J. Xu. Extending network lifetime for precision-constrained data aggregation in wireless sensor networks. In *IEEE Infocom*, Barcelona, Spain, April 2006.

[31] 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.

[32] 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.

[33] 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.

[34] J. Yoon, M. Liu, and B. Noble. Sound mobility models. In *ACM MobiCom*, San Diego, CA, Sept. 2003.

[35] W. Zhang and G. Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. In *IEEE Infocom*, Hong Kong, March 2004.

[36] 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.

Appendix: Analysis of 2-Dimensional Random Walk Model

Consider a 2-dimensional random walk model in which at each step, an object moves a distance of d in a direction uniformly distributed in $[0, 2\pi)$. Assume the object starts a random walk from point O. We divide the plane into a set of

rings of sufficiently small width Δ. As shown in Figure 12, ring 0 is enclosed by a circle centered at O with radius $\frac{1}{2}\Delta$, i.e., ring 0 contains all points that are within distance $\frac{1}{2}\Delta$ from O. For each $i \geq 1$, ring i is enclosed by two circles centered at O with radii $(i-\frac{1}{2})\Delta$ and $(i+\frac{1}{2})\Delta$, respectively, i.e., ring i includes all points that are $(i-\frac{1}{2})\Delta$ to $(i+\frac{1}{2})\Delta$ away from O.

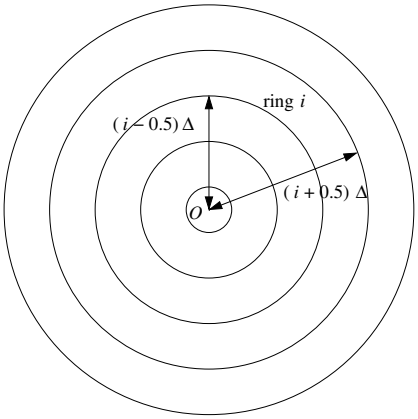


Fig. 12. A Set of Rings

Without loss of generality, we assume $d = k\Delta$, where k is an integer. Consider an object located in ring i. We approximate its distance to O as $\Phi = i\Delta$. Figure 13 shows that if the object moves in direction θ, its new distance to O is given by

$$\begin{aligned}\Phi' &= \sqrt{(d\sin\theta)^2 + (i\Delta + d\cos\theta)^2} \\ &= \sqrt{d^2 + (i\Delta)^2 + 2di\Delta\cos\theta} \\ &= \Delta\cdot\sqrt{k^2 + i^2 + 2ki\cos\theta}.\end{aligned}$$

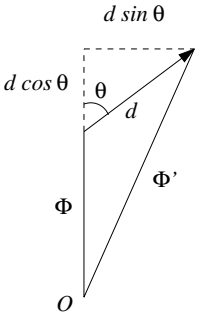


Fig. 13. Distance Computation

The object is located in ring j after the move if and only if

$$(j-\frac{1}{2})\Delta \leq \Delta\cdot\sqrt{k^2+i^2+2ki\cos\theta} \leq (j+\frac{1}{2})\Delta,$$

or equivalently,

$$\frac{(j-\frac{1}{2})^2 - i^2 - k^2}{2ki} \leq \cos\theta \leq \frac{(j+\frac{1}{2})^2 - i^2 - k^2}{2ki}.$$

If $j < |i-k|$ or $j > i+k$, the above condition is impossible to satisfy. In this case, the transition probability from ring i to j is given by 0.

If $j = |i-k|$, the new distance can be in the range $[j\Delta, (j+\frac{1}{2})\Delta]$ only. Due to symmetry, only the range $[0,\pi]$ needs to be considered for θ. Thus, the transition probability from ring i to j is given by

$$\frac{1}{\pi}\cdot\Big(\arccos\frac{j^2-i^2-k^2}{2ki} - \arccos\frac{(j+\frac{1}{2})^2-i^2-k^2}{2ki}\Big).$$

If $j = i+k$, the new distance can be in the range $[(j-\frac{1}{2})\Delta, j\Delta]$ only. Hence, the transition probability from ring i to j is given by

$$\frac{1}{\pi}\cdot\Big(\arccos\frac{(j-\frac{1}{2})^2-i^2-k^2}{2ki} - \arccos\frac{j^2-i^2-k^2}{2ki}\Big).$$

If $|i-k| < j < i+k$, the new distance can be in the range $[(j-\frac{1}{2})\Delta, (j+\frac{1}{2})\Delta]$. Therefore, the transition probability from ring i to j is given by

$$\frac{1}{\pi}\cdot\Big(\arccos\frac{(j-\frac{1}{2})^2-i^2-k^2}{2ki} - \arccos\frac{(j+\frac{1}{2})^2-i^2-k^2}{2ki}\Big).$$

We model each ring as a state and represent the object location during a random walk as a probability vector $[p_0, p_1, p_2, \ldots]$, where p_i is the probability that the object is located in ring i. Starting from the vector $[1, 0, 0, \ldots]$ (i.e., the object starts a random walk from point O), the vector after each move can be computed iteratively with the above transition probabilities. It is then easy to calculate the probability of the object's first moving beyond a given ring i (i.e., distance $x = (i+\frac{1}{2})\Delta$ from O) at each step. In this way, the *average number of steps* t the object takes to first move beyond x away from O can be derived numerically.

Figure 14 shows the simulation results, where the x-axis represents the normalized distance x/d from the starting point O, and the y-axis represents the ratio of t to $(x/d)^2$. As seen from Figure 14, when x is beyond a few times d, the ratio approaches a constant 1. Therefore, t can be approximated by $(x/d)^2$.

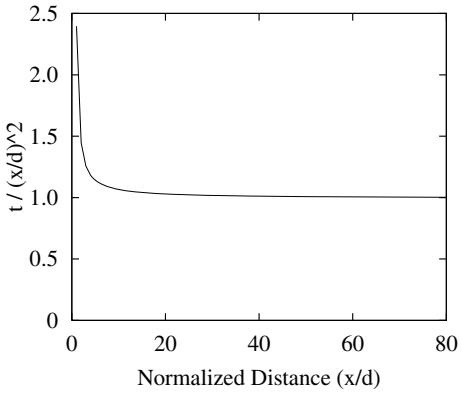


Fig. 14. Simulation Results of a Random Walk