Computer Communications 33 (2010) 1937–1948

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

ALM: An adaptive location management scheme for approximate location queries in wireless sensor networks

Lo-Yao Yeh, Chen-Che Huang, Cheng-En Wu, Jiun-Long Huang *

Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, ROC

A R T I C L E   I N F O

Article history:
Received 29 October 2008
Received in revised form 28 April 2010
Accepted 11 June 2010
Available online 20 June 2010

Keywords:
Approximate location queries
Location management
Wireless sensor networks

A B S T R A C T

Location management is an important issue for object tracking applications of wireless sensor networks. Since locations obtained by most positioning techniques are inherently imprecise, users may send approximate location queries with precision constraints to trade for energy consumption of sensor nodes. Therefore, we propose an Adaptive Location Management scheme called ALM to process approximate location queries. ALM employs a two-tier storage architecture (i.e., centric storage node and local storage node) to facilitate approximate query resolving and to reduce the energy consumption of sensor nodes. We propose a storage node relocation and replication mechanism, which can create, remove replicas of storage nodes and adjust the positions of storage nodes according to the location queries and updates. As such, the number of forwarding location update and query messages is reduced by the proposed storage node relocation and replication mechanism, thereby conserving more energy. The experimental results show that compared with EASE (Xu et al. (2008) [18]), ALM is able to reduce the number of transmission messages and the energy consumption of sensor network, and prolong the network lifetime.

1. Introduction

A sensor network is a wireless network composed of sensor nodes. A sensor node is a small device which has sensing, computing and communicating capabilities. It collects environmental data, processes the sensed data, and transmits them to other sensor nodes via wireless channels. In data transmission, the sensor node can act as a storage storing the data or as a router forwarding messages [1]. Object tracking is an important application of wireless sensor networks, such as monitoring wild animals or tracking military tanks [12,22]. In object tracking applications, sensor nodes should detect the position of a target[1] and store its location in a designated storage node. When the target moves, sensor nodes are responsible for tracking its location for users to query later. There are many applications of object tracking. These applications involve detecting, positioning, data transmitting in sensor networks and have attracted extensive attention in recent years [6,7].

Many efficient schemes for object tracking in wireless sensor networks have been proposed in the literature. These works can be classified into two categories. The first category focuses on detecting and positioning, including the technologies to position the target's location or to design a schedule for sensing nodes to turn into sleep mode when the target is not in their sensing ranges [3,21]. The second category is location management [9–11,14,17]. The location information of a target is gathered in a specific node for querying and updating later. Basically, there exists a tradeoff between the cost of location queries and location updates. Therefore, the challenge of location management is to design a proper location query and location update mechanism to strike a balance between the costs of location queries and updates.

One widely adopted approach in location management is the sink-based storage scheme. A sink is an external node to store the data updated by sensing nodes and to return the answers to querying nodes. The position of the sink is known for all nodes in the sensor network. An alternative approach is the in-network based storage scheme, which means that the data are stored in a or some sensor node(s) in the sensor network. Data-centric storage scheme, abbreviated as DCS, [14] is a well-known in-network based storage scheme. In DCS, the events are classified and named. When detecting an event, a sensor node applies a hash function to calculate the position of the corresponding storage node and then asks the storage node to store the event. The sink-based scheme is suitable for the case that query sources are not in the sensor network, while the in-network based scheme is suitable for the case that query sources are in the sensor network.

Since the energy conservation is the most important issue, tolerable errors in query answers are acceptable as long as query answers are sufficiently precise. In such applications, users send their

* Corresponding author. Tel.: +886 3 5712121.
E-mail addresses: lyyeh@cs.nctu.edu.tw (L.-Y. Yeh), cchuang.cs95g@nctu.edu.tw (C.-C. Huang), cewu@cs.nctu.edu.tw (C.-E. Wu), jlhuang@cs.nctu.edu.tw (J.-L. Huang).

[1] A target is an object tracked by the sensor network. In this paper, 'target' and 'object' are used interchangeably.

0140-3664/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.06.012

queries with precision constraints, called *approximate queries* [4]. Several studies indicated that the energy consumption could be greatly reduced when users accept approximate answers [4]. Due to the inherent imprecision of current positioning techniques, locations with little imprecision are acceptable in object tracking applications. Thus, an Energy-conserving Approximate StoragE scheme, called EASE [18], was proposed to reduce the energy consumption of the location management in sensor networks by utilizing *approximate location queries*. EASE stores location data of targets in a two-tier architecture. For each target, the high-precision location data are stored in a sensor node called *local storage node* which is close to the target, while the low-precision location data are stored in another sensor node called *centric storage node*. Location updates of a target are only sent to the corresponding centric storage node unless the target moves too far. Each location query asking the target's location is first sent to the centric storage node of the target. This query can be answered by the centric storage node if the low-precision location data can fulfill the precision constraint of the location query. Otherwise, if the location data stored in the centric storage node are not precise enough, the query will be forwarded to the local storage node to answer this query.

Although having been shown to be able to reduce the energy consumption [18], we argue that the energy consumption can be further reduced by improving the following two characteristics:

1. *The position of the centric storage node is fixed.*
   It is possible that the node which submits a location query is far from both the centric storage node and the local storage node. In such situation, processing the location query will consume many transmission messages in forwarding the query to the centric storage node and to the local storage node. In addition, processing one location update will also consume many transmission messages. Consider the scenario shown in Fig. 1(a). For simplicity, we only consider centric storage node here. Both one location query and one location update consume four messages. If the position of the centric storage node is able to be adjusted, we can move the centric storage node closer to the user and the target to reduce the number of transmission messages. In Fig. 1(b), since the centric storage node is moved toward the user and the target, the number of transmission messages can be cut down from 8 to 5. Therefore, it is beneficial to design a relocation mechanism for adjusting the position of centric and local storage nodes. In addition, the energy consumption can also be reduced due to the fact that the energy consumption in data communication is much more than that in data sensing and processing [1].
2. *Each target has only one centric storage node and one local storage node.*
   We first consider centric storage nodes. Since one target has only one centric storage node, the centric storage node has to process all location queries (no matter whether the centric storage node can answer them or not), thereby increasing the energy consumption of the centric storage node. Local storage nodes also suffer from the same problem. Thus, a replication mechanism to distribute the load of centric and local storage nodes is necessary.

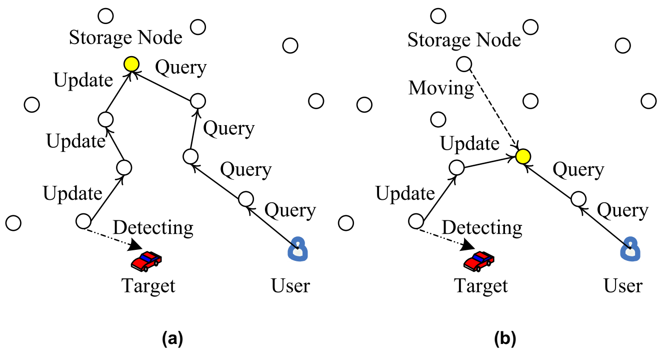

Fig. 1. The benefit of storage node relocation.

In view of above, we propose an Adaptive Location Management scheme, abbreviated as ALM, for approximate location queries in wireless sensor networks. Similar to EASE, ALM employs a two-tier storage architecture to store location data. To reduce the energy consumption, ALM is equipped with a storage node relocation and replication mechanism, which is able to adjust the positions of storage nodes and introduce new storage nodes according to the sources of location queries and updates. To provide the flexibility of the replication mechanism, the storage node splitting and merging mechanisms are also developed to avoid the redundant transmission messages. To evaluate the performance of ALM, several experiments are conducted. Compared with EASE, the experimental results show that ALM is able to reduce the number of transmission messages and to decrease the access latencies of location queries. Moreover, it is observed that ALM can reduce the energy consumption of sensor nodes and prolong the network lifetime.

The rest of this paper is organized as follows. Section 2 reviews the related work. The system overview and the details of ALM are presented in Section 3. The performance evaluation is presented in Section 4. Finally, Section 5 draws a conclusion.

2. Related work

Location management [10,15,18] consists of two basic procedures: location update and location query. The location of target is updated when the target moves or the user queries the interested target's location. When the object moves into the sensing field, the sensor nodes sense the object, determine the location of the object, and store the location of the object in a storage node. When a user wants to know the location of a target, it sends a location query message to the storage node to query the location of the target.

In object tracking applications of sensor networks, sensing and communicating are two major energy-consuming operations of sensor nodes. As a result, the methods proposed for object tracking are divided into two categories. One is trying to reduce the energy consumption in sensing operations. In some studies [3,21], there are two major operation modes of sensor nodes, namely, sleeping mode and sensing mode. Initially, all sensor nodes turn into sleeping mode. The sensor nodes wake up and sense the target only when the target is around them. They predict the target's moving trajectory and notify each sensor node that the target will move into its sensing range. The sensor nodes receiving the notify message should wake up and turn to the sensing mode. Thus, only sensor nodes near the object will be in the sensing mode and the other sensor nodes will remain in sleep mode to conserve energy.

The other category is to conserve the energy consumption in communicating operations. The neighboring sensor nodes of the target sends a location update message when the target moves or the user sends a location query message to obtain the location of the target of interest. Since all these two kinds of messages are sent to the location server, the position of the location server greatly influences the number of transmission messages. The traditional approach is using a fixed external server, or called the sink, and all messages are routed to this fixed server. This approach is suitable for applications where the queries are from external networks [9,11]. Most of these works dedicated to decreasing the number of update messages. In some schemes [9,11], the hierarchical tree structure is used to route update messages at some internal nodes. Then the sink has to forward query messages to those internal nodes to get information. Although it reduces the number of update messages, some extra query messages are produced. However, when the queries are generated from the sensor networks, both query and update messages have to be taken into account. Thus, some methods distribute the location servers in the sensor networks. In the scheme proposed by Li et al. [10], the area covered by the sensor network is partitioned into a hierarchy of grids. In each level of the grids, a node is assigned to several location servers for fault-tolerance and load-balance. In the scheme proposed by Wu [17], each target is associated with a geographic area called the virtual home region, abbreviated as VHR, of the target. All sensor nodes within the VHR serve as the location servers of the target at a probability.

Several studies focused on approximate queries [19]. An approximate location query is a tuple $\langle o_{id}, e\rangle$ where o_{id} is the identifier of the target object and e is the tolerable error in the object location. The data can be divided into different accuracy levels. The *lower accuracy data* can be stored at the home server and the *higher accuracy data* are stored at the local server. Some approximate queries with low-precision constraints can be answered by the home server, and the local server should only process approximate queries with high-precision constraints. Thus, the load of the home server and local server can be balanced and the number of transmission messages can be reduced. The study proposed by Xu et al. [18] indicated that using approximate queries can increase the benefit of hierarchical location servers. In the study [18], home servers and local servers are called *centric storage node* and *local storage node*, respectively. When moving within the acceptable range, an object only updates it's location stored its local storage node which is near the object. When moving beyond the approximate radius, the object will update the location data stored in both the centric and local storage nodes. Such a design is able to reduce the number of transmission messages while location queries and updates occur.

3. ALM: adaptive location management scheme

The details of ALM are given in this section. For better readability, we first describe the basic version of ALM without the storage node relocation and replication mechanism in Section 3.1. Then, Section 3.2 elaborates the proposed storage node relocation and replication mechanism. To cooperate with the proposed storage node relocation and replication mechanism, the basic version requires some revisions on processing location updates and approximate location queries, and the revisions are described in Section 3.3.

3.1. The basic system

In the basic version of ALM, there is one centric storage node and one local storage node for each target. In this paper, we use Geographic Hash-Table, named GHT method proposed by Ratnasamy et al. [14], to determine the position of the centric storage node of each target. A centric storage node is responsible of storing the low-precision location data of the corresponding target. The errors of the locations stored in the centric storage node are of bound by an approximate radius r. It means that for an arbitrary object o, if the low-precision location of the object o is p, the distance between the precise location and the low-precision location of the object o is less than or equal to r. That is, the object o is within the circle of the radius r centering at p. Any geographic routing protocol (e.g.,

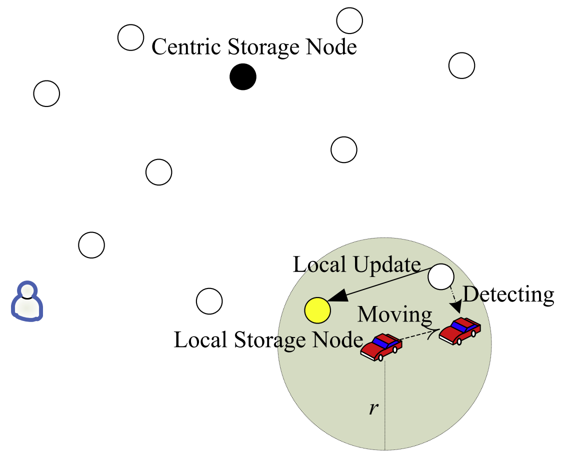

Fig. 2. Location update procedure.

GPSR [8] or BLR [5]) can be employed as the underlying routing protocol of ALM.

3.1.1. Location update and local storage node handoff

Initially, the node first detecting the object o serves as the local storage node, and sends a registration message to the centric storage node of the object o. The registration message includes the id of the local storage node, and the current location of the object o. All neighboring nodes in the approximation radius area are notified by the local storage node. If the object o moves within its current approximate radius area, the node first detecting the object o performs the location update procedure; otherwise, the node first detecting the object o performs the local storage node handoff procedure.

- Location update

As shown in Fig. 2, the location update process of the basic system consists of the following steps.

Step 1: The node first detecting the object o sends a local update message to the object o's local storage node. The local update message contains the current location of the object o.

Step 2: When receiving the local update message, the object o's local storage node stores the location information of the object o into its local storage.

Note that, since the object o only locally moves within the range of its current approximation radius area, the centric storage node does not need to update the location of the object o.

- Local storage node handoff

The local storage node handoff procedure is invoked while the object o goes out of its current approximate radius area. The local storage node handoff procedure is to (1) store the object o's location in the new local storage node, (2) remove the location information stored in the old local storage node, and (3) update the location information stored in the centric storage node. As shown in Fig. 3, the local storage node handoff procedure consists of the following steps.

Step 1: If the object o moves out of its current approximate radius area, the node first detecting the object o takes over the object o's new local storage node, and creates a record to store the object o's current location.

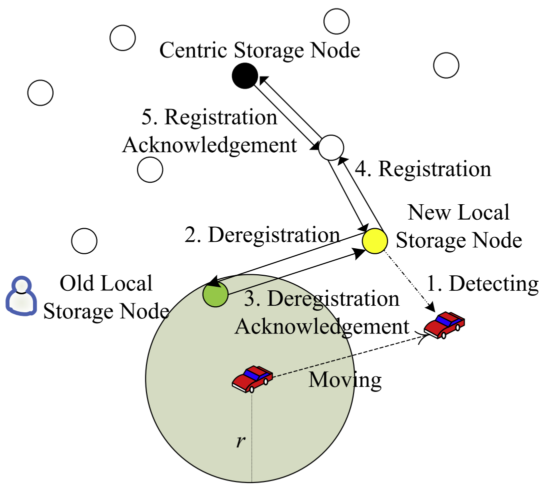

Fig. 3. Handoff procedure.

Step 2: The new local storage node sends a deregistration message to the object o's old local storage node.[2]

Step 3: After receiving the deregistration message, the object o's old local storage node deletes the object o's record. And the old local storage node sends a deregistration acknowledgment message to the new local storage node.

Step 4: Then, the new local storage node delivers a registration message with its location to the object o's centric storage node to notify the object o's centric storage node to change the location information of the local storage node.

Step 5: The centric storage node first updates the location of the object o and the location of the object o's new local storage node, and then replies a registration acknowledgment message to the object o's new local storage node. When the new local storage node receives the registration acknowledgment message, the local storage handoff procedure is finished.

3.1.2. Location query

When a user u wants to query the location of the object o, the location query procedure will be invoked. As shown in Fig. 4, the location query procedure of the basic system consists of the following steps.

Step 1: The user u first calculates the location of the centric storage node of the object o by GHT. Then, user u issues an approximate location query message to the centric storage node. The approximate location query message contains the id of the object o and the precision constraint e.
When receiving the query message, the centric storage node first compares the precision constraint e and the approximate radius r. If $e \geqslant r$, the centric storage node executes Step 2-1. Otherwise, Step 2-2 will be executed.

Step 2-1: $e \geqslant r$ means the low-precision location stored in the centric storage node is accurate enough for the received approximate location query. Thus, the centric storage node will answer this query message by responding the stored low-precision location of the object o to the user u, and then the location query procedure is finished.

[2] If the location of the old location storage node is unknown, then the new local storage node can query the location information from the object o's centric storage node.

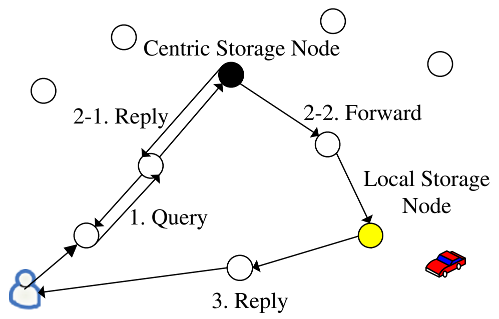

Fig. 4. Location query procedure.

Step 2-2: When the low-precision location is not accurate enough (i.e., $e < r$), the centric storage node forwards this query message to the object o's local storage node.

Step 3: When receiving the query message from the centric storage node, the local storage node of the object o returns the stored high-precision location of the object o to the user u, and then the location query procedure is finished.

3.2. Storage node relocation and replication

As observed in Fig. 1, moving storage nodes closer to users or objects is able to reduce the number of transmission messages. Thus, we propose a storage node relocation and replication mechanism which consists of three operations: *storage node moving*, *storage node splitting* and *storage node merging*. The storage node moving operation allows to adjust the positions of centric and local storage nodes. Moreover, when necessary, ALM will replicate the content stored in storage nodes to some other neighboring nodes by the storage node splitting operation and remove replicas by the storage node merging operation.

In essence, ALM consists of two phases: *statistics collection phase* and *adaptation phase*. Initially, each node is in the statistics collection phase, and switches to the adaptation phase every t_{check} seconds. In the statistics collection phase, each sensor will collect some statistics such as how many location query and update messages are received from neighboring nodes. In the adaptation phase, each storage node will evaluate whether to execute storage node relocation or replication operations. The storage node will turn back to the statistics collection phase after one of the three operations is performed. Other nodes without executing any operations will switch back to the statistics collection phase directly. The details of these operations are described in the following subsections.

3.2.1. Storage node moving

Consider the scenario shown in Fig. 5. Let $N = \{n_1, n_2, \ldots, n_{|N|}\}$ be the set of neighboring nodes of a node n_0. Suppose that the node n_0 receives $T_{n_0}(n_i)$ messages (location query and location update messages) from its neighbor n_i, and receives $T_{n_0}(n_0)$ messages di-

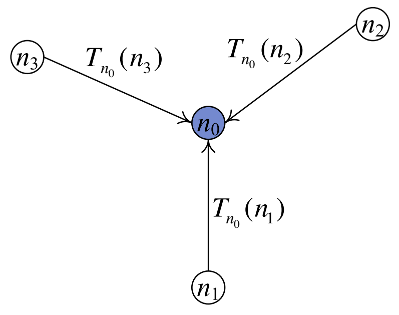

Fig. 5. Message considerations for the storage node moving operation.

rectly from users. If we move the storage node from the node n_0 to a node n_i, all messages from n_i to n_0 can be saved. However, the node n_0 has to forward all messages received from users and from all its neighbors except n_i to n_i.[3] Thus, such a moving operation will produce extra $\sum_{\forall n_j \in (N \setminus \{n_i\})} T_{n_0}(n_j) + T_{n_0}(n_0)$ messages as well as the update cost, denoted by $U_{n_0}(C)$, of notifying the centric storage node of the movement.[4] To evaluate the number of transmission messages, the message reduction, denoted by $R_{move}(n_i)$ is defined as

$$R_{move}(n_i) = T_{n_0}(n_i) - \left(\sum_{\forall n_j \in (N \setminus \{n_i\})} T_{n_0}(n_j) + T_{n_0}(n_0) \right) - U_{n_0}(C) \qquad (1)$$

It is valuable to perform the storage node moving operation if $R_{move}(n_i)$ is larger than zero. However, in order to avoid the ping-pong effect, a moving operation is executed only when $R_{move}(n_i)$ is larger than or equal to a pre-specified moving threshold δ_{move}. ALM decides to move the storage node from n_0 to its neighbor n_k of the maximal message reduction. Here, we define our storage node moving formulation:

$$n_k = \begin{cases} \varnothing & \text{if } \max(R_{move}(n_j)) < \delta_{move} \\ \{n_l \mid R_{move}(n_l) \\ \quad = \max(R_{move}(n_j))\} & \text{otherwise} \end{cases} \qquad \forall n_j \in N$$

Finally, the storage node moving operation is summarized in Algorithm 1 (StorageNodeMoving).

Algorithm 1 (StorageNodeMoving)

```
• Output: TRUE is returned when the storage node is moved.
  Otherwise, FALSE is returned.
1.  $MaxR \leftarrow 0$
2.  $TargeNode \leftarrow NULL$
3.  for each neighboring node, say $n_l$, of node $n_0$ do
4.     Let $R_{move}(n_l)$ be the result of Eq. (1)
5.     if $R_{move}(n_l) > MaxR$ and $R_{move}(n_l) > \delta_{move}$ then
6.        $MaxR \leftarrow R_{move}(n_l)$
7.        $TargetNode \leftarrow n_l$
8.     end if
9.  end for
10. if $TargetNode \neq NULL$ then
11.    Move the storage node from node $n_0$ to node
       TargetNode
12.    return TRUE
13. else
14.    return FALSE
15. end if
```

Example 1. Consider the example shown in Fig. 6 where the node n_0 is a storage node. Suppose that the node n_0 does not directly receive any message from users (i.e., $T_{n_0}(n_0) = 0$) and assume the moving threshold (δ_{move}) and update cost ($U_{n_0}(C)$) be 2. By Eq. (1), the message reduction of moving the storage node from n_0 to n_1 is $R_{move}(n_1) = T_{n_0}(n_1) - (T_{n_0}(n_2) + T_{n_0}(n_3) + T_{n_0}(n_0)) - U_{n_0}(C) = 12 - (4 + 3 + 0) - 2 = 3$. Similarly, $R_{move}(n_2)$ and $R_{move}(n_3)$ are -13 and -15, respectively. Since $R_{move}(n_1)$ is the maximum number of reduction messages and is larger than δ_{move} (i.e., 2), ALM moves the storage node from n_0 to n_1.

[3] Note that it is possible that some neighboring nodes of the node n_0 can directly deliver the message to n_i if the radio range of the neighboring nodes reaches n_i. However, it depends on the topology of sensor nodes. For simplicity, in this paper, we concern the worst case that each message has to be relayed by the node n_0.

[4] If the movement happens on centric storage nodes, the movement information is recorded in the object o's initial centric storage node for redirection.

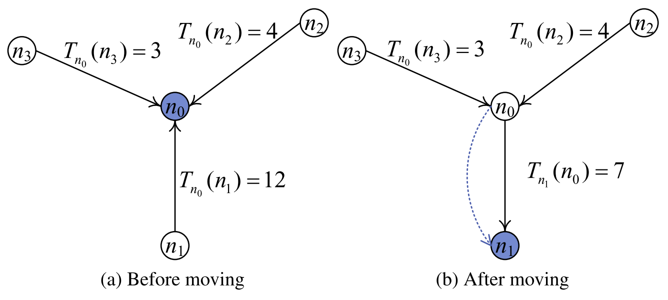

Fig. 6. Storage node moving.

3.2.2. Storage node splitting

ALM can split a storage node into several replicas when necessary. Read-one-write-all model [13] is employed to keep the consistency of all replicas. A local update of the object o is completed if and only if all object o's replicated storage nodes have been updated. A location query to the object o is completed when the user receives the response from one of the object o's storage node. The advantages of several replicas are twofold. First, the number of location query message transmissions is reduced. Second, the load of storage nodes can be alleviated. However, more replicas may lead to more update messages. To facilitate the storage node splitting operation, we should keep track of the numbers of location query and update messages. Let $N = \{n_1, n_2, \ldots, n_{|N|}\}$ be the set of neighboring storage nodes of a node n_0. $Q_{n_0}(n_i)$ denotes the number of location query messages that the node n_0 received from a node n_i for $1 \leqslant i \leqslant |N|$, $U_{n_0}(n_0)$ denotes the number of location update messages generated by the node n_0 itself, and $U_{n_0}(n_i)$ denotes the number of location update messages that the node n_0 received from the node n_i for $1 \leqslant i \leqslant |N|$, respectively. Fig. 7 illustrates an instance of the storage node splitting operation. If we split the storage node n_0 to a node n_i, the location queries from the node n_i to the node n_0 can be reduced. However, the node n_0 has to forward all received location updates to the node n_i, which increases some extra location update messages. Therefore, it is beneficial to perform storage node splitting from n_0 to n_i if the message reduction, denoted as $R_{split}(n_i)$, is larger than or equal to the pre-defined splitting threshold, denoted by δ_{split}, where $R_{split}(n_i)$ is defined as

$$R_{split}(n_i) = Q_{n_0}(n_i) - \left(\sum_{\forall n_j \in N} U_{n_0}(n_j) + U_{n_0}(n_0) \right) \qquad (2)$$

ALM splits the storage node n_0 to the neighbor nodes n_k if the message reduction $R_{split}(n_k)$ is larger than splitting threshold δ_{split} that is used to avoid the ping-pong effect. The formulation of storage node splitting is defined as follows:

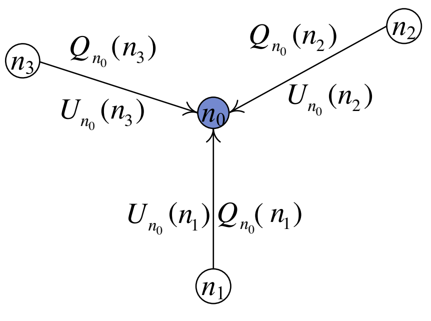

Fig. 7. Message consideration for the storage node splitting operation.

$$n_k = \begin{cases} \varnothing & \text{if } \max(R_{split}(n_j)) < \delta_{split} \\ \{n_l | R_{split}(n_l) \geqslant \delta_{split}\} & \text{otherwise} \end{cases} \quad \forall n_j \in N$$

Finally, the storage node splitting operation is summarized in Algorithm 2 (StorageNodeSplitting). Note that to reduce the number of transmission messages, the storage node splitting operation does not send the location notification message to the centric storage node since the centric storage node can still query the high-precision location of the object o from the original local storage node.

Algorithm 2 (StorageNodeSplitting)

```
• Output: TRUE is returned when the storage node is split.
  Otherwise, FALSE is returned.
1.  TargeNode[ ] ← NULL
2.  for each neighbor non-storage node, say $n_l$, of the node $n_0$
    do
3.    Let $R_{split}(n_l)$ be the result of Eq. (2)
4.    if $R_{split}(n_l) \geqslant \delta_{split}$ then
5.      TargetNode[ ] ← $n_l$
6.    end if
7.  end for
8.  if TargetNode[ ] ≠ NULL then
9.    Split the storage node from the node $n_0$ to nodes
      $n_l \in$ TargetNode[ ]
10.   return TRUE
11. else
12.   return FALSE
13. end if
```

Example 2. Consider the example shown in Fig. 8 where the node n_0 is a storage node. Suppose that δ_{split} is 2. If the storage node is splitted from the node n_0 to the nodes n_2 and n_3, the location query messages from the nodes n_2 and n_3 to the node n_0 can be reduced. However, the node n_0 has to forward the received location update messages to the nodes n_2 and n_3 for the consistency. By Eq. (2), the message reduction of splitting the node n_0 to the node n_2 is $R_{split}(n_2) = Q_{n_0}(n_2) - U_{n_0}(n_1) = 6 - 3 = 3$. Similarly, $R_{split}(n_1)$ and $R_{split}(n_3)$ are -1 and 5, respectively. As a result, ALM splits the storage node from the node n_0 to the nodes n_2 and n_3.

3.2.3. Storage node merging

Although multiple replicas allow to reduce the number of location query messages, the expense of multiple replicas is to increase the number of location update messages. To mitigate this problem, storage node merging is developed to reduce the number of replicas when necessary. Let $N1 = \{n_1, n_2, \ldots, n_{|N1|}\}$ be the set of the neighboring non-storage nodes of a node n_0, and $N2 = \{n_1, n_2, \ldots, n_{|N2|}\}$ be the set of the neighboring storage nodes of the node n_0. As shown in Fig. 9, if the storage node n_0 is merged into a neighbor storage node $n_i \in N2$, the number of location update messages from the node n_i to the node n_0 is reduced. However, the node n_0 has to additionally forward all received location queries to the node n_i. Thus, it is valuable to merge the storage node n_0 into n_i if the message reduction of merging, denoted by R_{merge}, is larger than or equal to the pre-defined merge threshold, denoted as δ_{merge}, where R_{merge} is defined as

$$R_{merge}(n_i) = U_{n_0}(n_i) - \left(\sum_{\forall n_j \in N1 \setminus \{n_i\}} Q_{n_0}(n_j) + Q_{n_0}(n_0) \right) - U_{n_0}(C) \quad \forall n_i \in N2 \tag{3}$$

where δ_{merge} is used to avoid the ping-pong effect. ALM will decide to merge the storage node n_0 into the neighbor n_k of the most message reduction if there are multiple merging options. The formulation of the storage node merging is defined as:

$$n_k = \begin{cases} \varnothing \\ \quad \text{if } \max(R_{merge}(n_j)) < \delta_{merge} \\ \{n_l | R_{merge}(n_l) = \max(R_{merge}(n_j))\} \\ \quad \text{otherwise} \quad \forall n_j \in N1, \ \forall n_i \in N2 \end{cases}$$

Algorithm 3 (StorageNodeMerging)

```
• Output: TRUE is returned when the node merging is performed.
  Otherwise, FALSE is returned.
1.  MaxR ← 0
2.  TargeNode ← NULL
3.  for each neighbor storage node, say $n_l$, of node $n_0$ do
4.    Let $R_{merge}(n_l)$ be the result of Eq. (3)
5.    if $R_{merge}(n_l)$ > MaxR and $R_{merge}(n_l) > \delta_{merge}$ and $n_l$ is a
      storage node then
6.      MaxR ← $R_{merge}(n_l)$
7.      TargetNode ← $n_l$
8.    end if
9.  end for
10. if TargetNode ≠ NULL then
11.   Merge the storage node from the node $n_0$ into the node
      TargetNode
12.   return TRUE
13. else
14.   return FALSE
15. end if
```

Example 3. Consider the example shown in Fig. 10. Suppose that δ_{merge} and $U_{n_0}(C)$ are set to 2. If the storage node n_0 is merged into the neighboring storage node n_2, the location update messages from n_2 to n_0 can be saved. However, the node n_0 has to forward all received location queries to the node n_2. According to Eq. (3), the message reduction is $R_{merge}(n_2) = U_{n_0}(n_2) - (Q_{n_0}(n_1) + Q_{n_0}(n_3)) - U_{n_0}(C) = 12 - (4 + 3) - 2 = 3$ which is larger than δ_{merge}. Thus, ALM will merge the storage node n_0 into the storage node n_2.

In the adaptation phase, each storage node will execute algorithm StorageNodeRelocationReplication to determine whether to perform the storage node relocation or replication mechanism. According to Wolfson et al. [16], the storage node splitting procedure should have higher priority than the storage node moving and merging procedures. Since at most one procedure will be performed in each storage node, the algorithm StorageNodeRelocationReplication can be terminated once one of those three procedures is performed. Finally, the algorithm StorageNodeRelocationReplication is listed in Algorithm 4.

Algorithm 4 (StorageNodeRelocationReplication)

```
1.  if $n_0$ is a centric storage node or $n_0$ is a local storage node
    then
2.    if StorageNodeSplitting ( ) = TRUE then
3.      return
4.    else if StorageNodeMerging ( ) = TRUE then
5.      return
6.    else if $n_0$ is the only one centric storage node of the object o
      or $n_0$ is the only one local node of the object o then
7.      Perform StorageNodeMoving ( )
8.    end if
9.  end if
```

3.3. Revisions on processing approximate location queries and location updates

To seamlessly integrate the proposed storage node relocation and replication mechanism with basic system, some minor revisions are discussed in this section. When a user u submits a

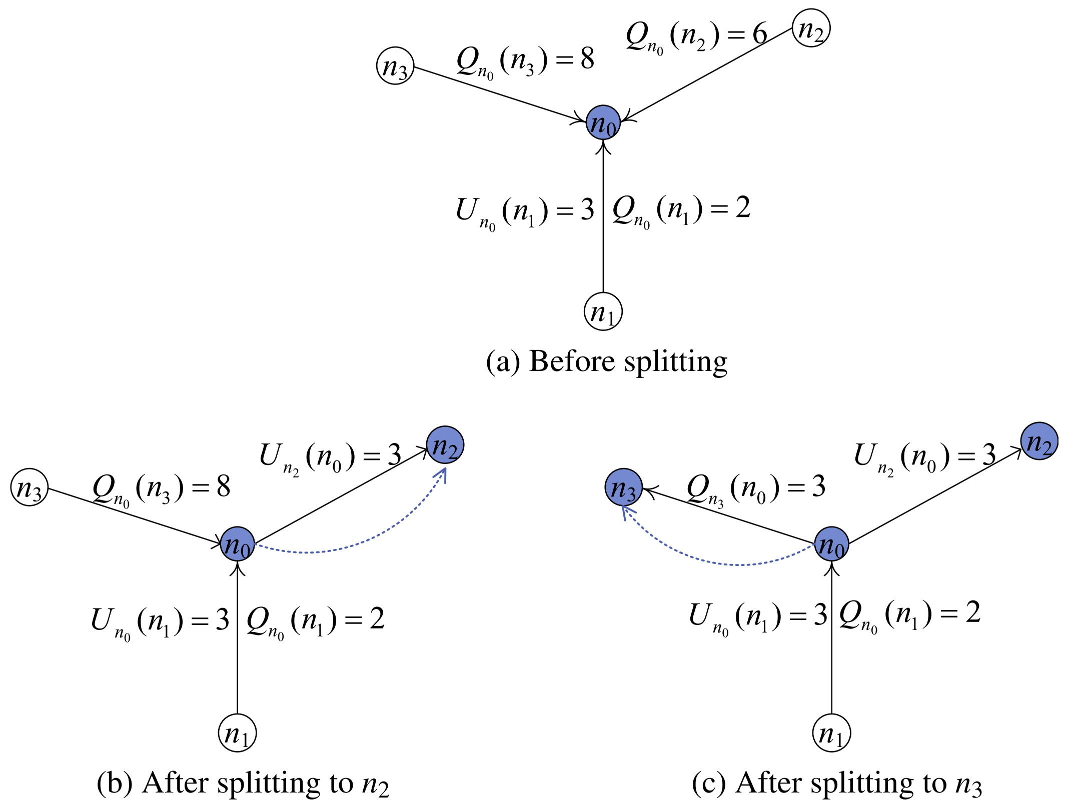

Fig. 8. Storage node splitting.

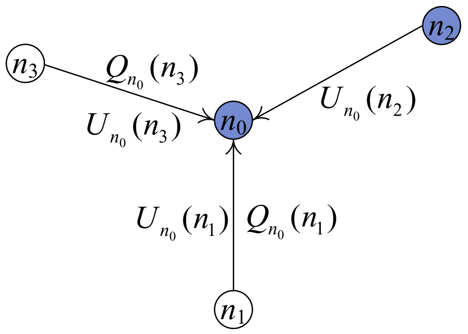

Fig. 9. Message consideration for the storage node merging operation.

location query for an object o, the query is sent toward the object o's centric storage node. If an intermediate node n_1 is a replica of the centric storage node of the object o, the node n_1 will act as the object o's centric storage node. If an intermediate node n_2 is a replica of the local storage node of the object o, the node n_2 will return the stored high-precision location data of the object o to the user u. In addition, according to the characteristics of storage node replication operations [16], the replicated storage nodes are connected. Once a replicated storage node receives an update message, it will process the update message as usual and forward the message to the other storage nodes for the data consistency. Next, we further discuss the details of some aspects as follows.

- Local storage node handoff

As mentioned in Section 3.1.1, the local storage node handoff procedure will be triggered when the object o moves out of its approximation radius area. It is possible that the local storage node of the object o is executing the storage node relocation and replication mechanism while the object o is moving to trigger a new handoff procedure. However, the handoff procedure will elect a new local storage node, and the storage node relocation and replication mechanism may do the same thing. To avoid the redundant relocation or replication, we set the local storage node elected by the handoff procedure with the highest priority, and the local storage nodes elected by the storage node relocation and replication mechanism will be considered as the replicated storage nodes.[5] In addition, if there are several replicated storage nodes generated by the storage splitting procedure, these replicated storage nodes remain after a new handoff procedure. But, it is anticipated that some replicated storage nodes may be merged in a short period to reduce the redundant query or update messages.

- Storage move and merge

Since the location of a storage node may be adjusted by the moving and merging operations, the location of the new storage node should be recorded to avoid the redundant transmission messages.

1. Local storage node
   When the location of a local storage node is changed by the storage node moving or merging operations, the old local storage node should send a location notification message to the centric storage node.
2. Centric storage node
   When the location of the centric storage node is changed by the storage node moving or merging operations, the initial centric storage node calculated by GHT method should keep a record to trace the location of the current centric storage node. There-

[5] The centric storage node can help to handle the priority setting.

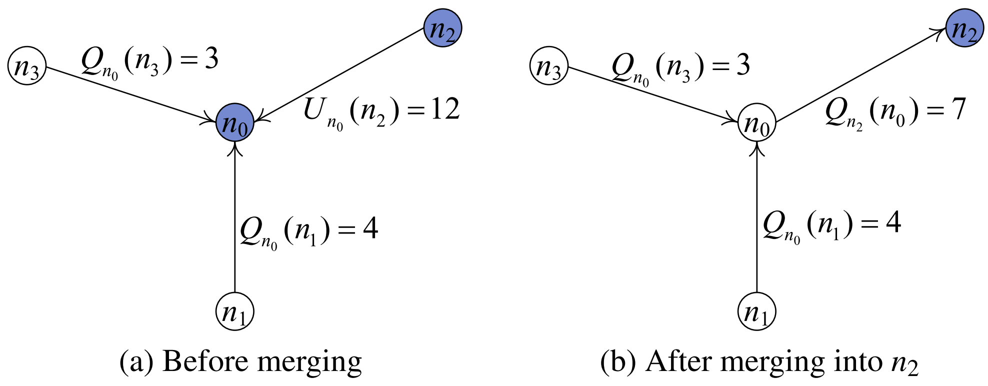

Fig. 10. Storage node merging.

fore, if a centric storage node is moved or merged, the old centric storage node should send a location notification message to initial centric storage node. As a result, the initial centric storage node can redirect the query and update messages to the current centric storage node.

4. Performance evaluation

4.1. Simulation model

We developed a simulator based on ns-2[6] to compare the performance of ALM and EASE [18]. GPSR [8] and BLR [5] are used as the routing protocols for both schemes. In GPSR, each forwarding node designates the neighbor closest to the destination as the next-hop. In BLR, a forwarding node broadcasts the data packet and those neighboring nodes closer to the destination set a timer to contend to be the forwarder. Thus, BLR is likely to have more than one neighbor to forward the packet if neighbors do not overhear others' forwarding. IEEE 802.11 DCF and the two-ray ground model are used as the underlying MAC protocol and the radio model, respectively. We deploy 100 sensor nodes in a 340×340 m^2 field. Similar to EASE [18], the field is divided into 100 34×34 m^2 grid cells and one sensor node is placed in the center of each cell. The energy consumption model and the parameters are set according to HEED [20]. We employ the linear model [2] as the mobility model in our simulation. The setting of each moving object includes the destination, the start moving time, and the speed. The destination of each moving object changes every 20 s. The object speeds are set between 1 and 5 m/s. The users issuing approximate queries are randomly distributed in the field. The tolerable error bounds of queries are uniformly distributed between 0 m and 50 m. By default, the query rate is set 10/s and the moving speed of objects is set to 2 m/s. Note that the query rate is defined as the total number of queries issued by querying nodes in the entire sensor network per second. Table 1 summarizes the system parameters used in the simulation.

Table 1
Simulation parameters.

Parameter	Setting
Field size	340×340 m^2
Number of nodes	100
Radio range	40 m
Sensor sampling rate	2 s
Checking period (t_{check})	2 s
Query rate	2–10/s
Tolerable error bound of queries	0–50 m
Speed of objects	1–5 m/s
Approximate radius	50 m
Initial energy of each node	5 J
E_{elec}	50 nJ
ϵ_{fs}	10 pJ/bit/m^2
ϵ_{ap}	0.0013 pJ/bit/m^4
Query message payload size	16 bytes
Update message payload size	32 bytes
Reply message payload size	32 bytes
Acknowledge message payload size	16 bytes
Query start time	10 s
Simulation time	500 s

4.2. Impacts on message size

4.2.1. Query rate

In this experiment, we evaluate the impact of query rate on message sizes produced by ALM and EASE. The query rate is ranging from 2/s to 10/s. From Fig. 11, we can see that the message reduction rate of ALM over EASE increases as the quare rate increases. Specifically, when using GPSR, the message reduction rate of ALM over EASE increases from 19.5% to 27.9% as the query rate is from 2/s to 10/s. On the other hand, with BLR, the message reduction rate increases from 4.6% to 20.6% with the query rate rising from 2/s to 10/s. When BLR is used, we observe that ALM and EASE produce more messages. The reason is that BLR establishes more than one forwarding paths due to the grid topology and shorter radio ranges of sensor nodes. As the query rate is high, ALM relocates and replicates centric and local storage nodes to those storage nodes closer to the users to reduce the number of query and

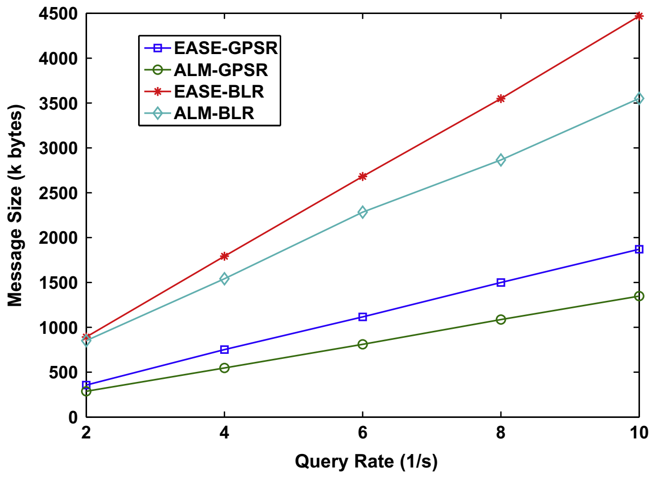

Fig. 11. Impact on query rate vs. message size.

[6] The network simulator – ns-2, http://www.isi.edu/nsnam/ns/.

reply transmissions. Due to the relocation and replication mechanism, the higher the query rate is, the more advantageous ALM is. As a result, ALM is very suitable for the environment with high query rate.

4.2.2. Moving speed

We investigate the impact of object moving speed on the message size in this experiment. As shown in Fig. 12, basically, the message sizes slightly increase as the moving speed increases except for EASE–GPSR. The increasing message sizes result from that higher moving speeds incur more update messages. With BLR, ALM steadily outperforms EASE due to the benefit of the proposed relocation and replication mechanism. The message reduction rate of ALM over EASE is between 20.54% and 11.39%. When GPSR is employed, we observe that ALM produces more messages than EASE as the object moving speed is larger than 5 m/s. Besides, different from the other combinations, EASE–GPSR produces less messages when the object moving speed is larger than 5 m/s. We explain this result as follows. In fact, the number of update messages of EASE increases as the moving speed increases. However, the increasing message size by update messages only is about 0.5% of the total message size. When the object moving speed is larger than 5 m/s, EASE suffers from severe message dropping problem. About 17.43–26.93% of the query messages or the corresponding reply message are lost that contribute the substantial reduction of the message size. The severe message dropping decreases the total message size greatly so that the slight rise of the message size by update messages becomes negligible. Note that because of the closer splitting and moving storage nodes, ALM only has 1.13% message loss with GPSR. Finally, we would like to mention that with BLR, the message dropping problem of EASE is mitigated thanks to the multiple paths.

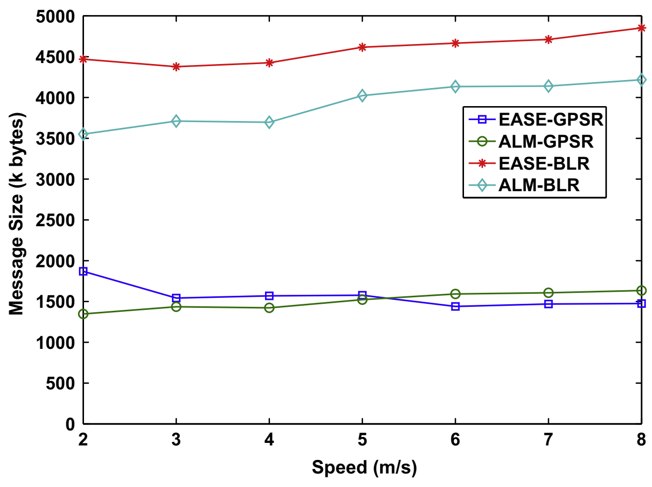

Fig. 12. Impact on object moving speed vs. message size.

4.2.3. Threshold and query distribution

We study the impact of threshold δ on message size in the uniform and bias query distributions. Fig. 13(a) shows the experimental result in the uniform distribution, while Fig. 13(b) depicts the result in the bias distribution. The threshold is varied from 2 to 10. First, we observe from Fig. 13(a) that with GPSR, ALM achieves better performance than EASE regardless of the threshold. As the threshold increases, the advantage of ALM over EASE becomes less significant. The reason is that the higher thresholds result in the proposed relocation and replication mechanism to be executed less. On the other hand, with BLR, ALM performs better than EASE when the threshold is less than 5. However, as the threshold is higher than 5, ALM suffers from worse performance than EASE. This is due to the fact that, when the number of replicated nodes is not sufficiently large, the multiple paths of BLR cause ALM to produce redundant query and reply messages from a closer replicated storage node and a farther storage node. This explanation is confirmed by that ALM experiences shorter latencies shown in the following. Next, we discuss the result with respect to the bias query distribution. To generate the bias distribution, we only allow the approximate queries submitted from the bottom-right 3×3 corner of the 10×10 grid field. As depicted in Fig. 13(b), when GPSR is used, the message reduction rate of ALM over EASE is about 40.5% for all thresholds. The message reduction rate of ALM over EASE further yields up to 50.4% with BLR except for $\delta = 8$. The substantial improvement in the bias query distribution is due to that the proposed relocation and replication mechanism easily leads splitting and moving storage nodes to be closer to the targets and querying nodes. In other words, update messages are quickly received and most of the query messages can be directly resolved by closer splitting and moving storage nodes.

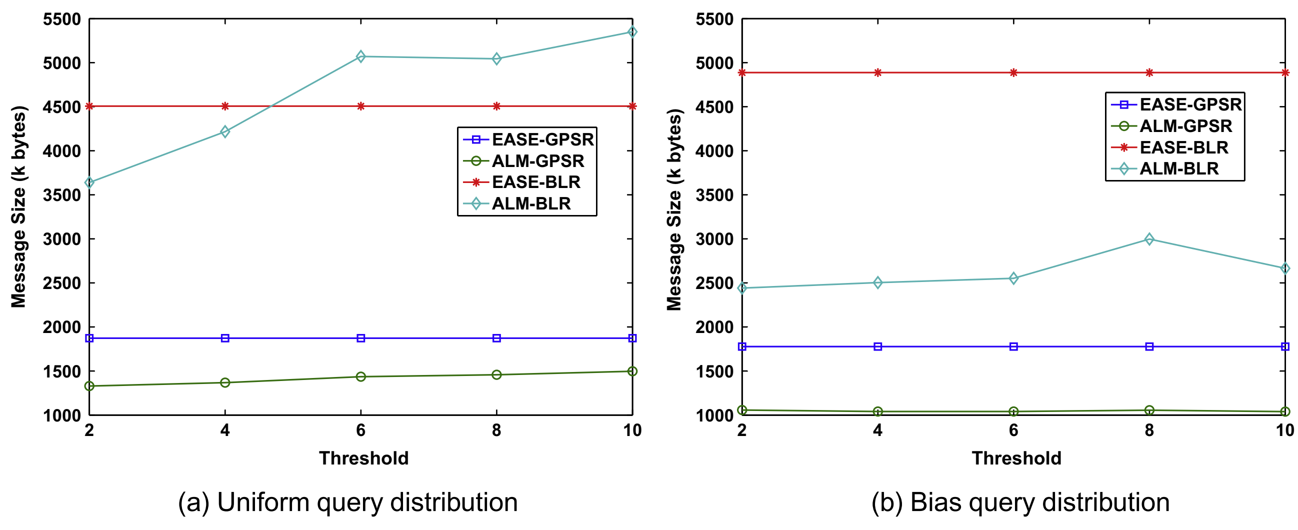

Fig. 13. Impact on moving threshold vs. message size.

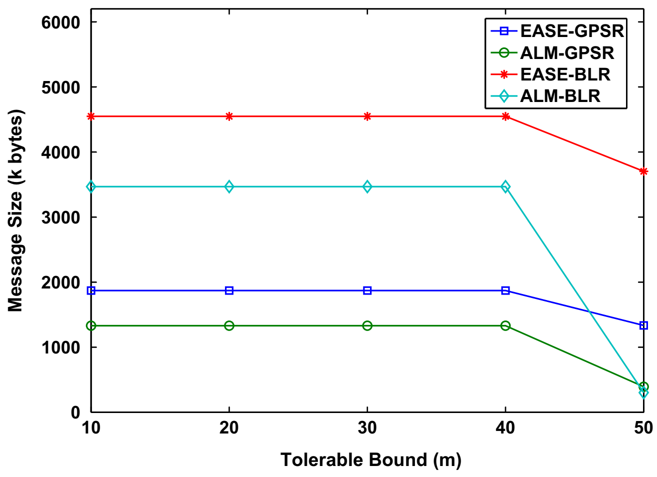

Fig. 14. Impact on tolerable error bound vs. message size.

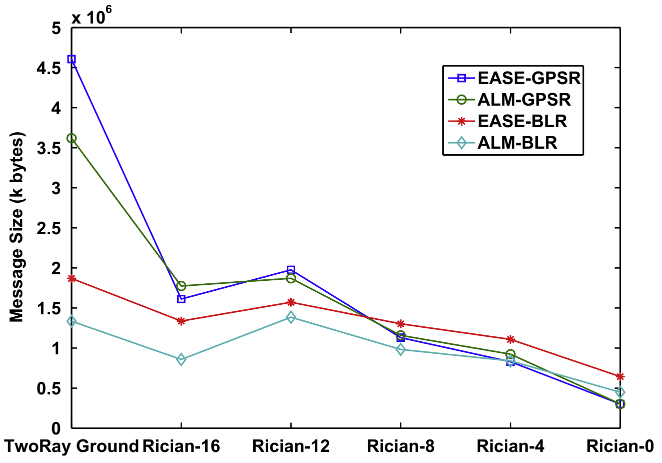

Fig. 15. Impact on channel model vs. message size.

4.2.4. Tolerable error bound

Fig. 14 shows the experimental result of ALM and EASE with tolerable error bound of queries varied. The tolerable error bound is varied from 10 m to 50 m. This experiment indicates the degree of energy consumption by approximate answers. We observe that the message sizes of ALM and EASE do not change when the tolerable error bound is between 10 m and 40 m. When the tolerable error bound is 50 m, the message sizes of ALM and EASE drop since the majority of approximate queries can be answered directly by centric storage nodes and their replicated and moving storage nodes. It is worthwhile to mention that compared with EASE, the message sizes of ALM drop significantly. This is because that most of the queries could be answered within a few hops by replicated and relocated storage nodes. In other words, the significant message size reduction results from that the number of query and reply messages are greatly reduced by closer storage nodes. Furthermore, the result verifies that the total energy can be substantially saved by allowing approximate queries.

4.2.5. Channel models

In this experiment, we examine the impact of channel model on message size of ALM and EASE. In addition to the default two-ray ground model, we employ the Rician-*K* fading model. In the Rician-*K* fading model, the *K*-factor is defined as the ratio of signal power in the dominant path over the scattered ones. The smaller the *K* is, the more scattered the signal power is and the higher the bit error rate is. The Rayleigh fading is a special case of the Rician-*K* fading model with *K* equal to 0. Note that in the ns-2 simulation, the radio range of a sensor node is determined by the corresponding transmission power under the two-ray ground model. Thus, the same transmission power under the Rician-*K* model has the shorter radio range. For fairness, we decide to use the same transmission power under different channel models. From Fig. 15, it can be seen that the message sizes all decrease as the channel condition becomes more adverse. Adverse channel conditions give rise to message size reduction due to severe message dropping. ALM is more resilient to error-prone channel conditions than EASE. For example, under the Rician-4 channel model, ALM receives 3384 query results with total message size 840,192 bytes, but EASE only obtains 2911 ones with total message size 1,108,704 bytes. With BLR, due to the unreliable broadcasting, both ALM and EASE are impaired by the substantial message dropping problem, which causes the message size reduction. Finally, when the Rician-12 channel model is used, there is a spike in the message sizes of all schemes. This is because the Rician-16 channel model suffers from shorter transmission ranges causing severer message dropping. However, when the value of *K* decreases, scattered signal strengths allow the transmissions beyond the radio range area, thereby increasing the delivering probability. The advantage of scattered signal strengths becomes less substantial as the value of *K* further decreases, which also increases the bit error rate significantly. To sum up, the scattered signal strengths and bit error rate are the reason of the spike in the message size.

4.3. Message distribution

The distribution of all types of messages is shown in Fig. 16. The update messages include local updates and forwarding updates. It can be seen that ALM produces fewer query messages and reply messages than EASE does. Since ALM replicates storage nodes and move storage nodes closer to query nodes, the queries could be resolved and the answers will be replied by closer storage nodes. In this experiment, for the counts of query and reply messages, the reduction rates of ALM over EASE are 50.1% and 11.8% using GPSR, respectively. On the other hand, when using BLR, the reduction rates of ALM over EASE are 44.2% and 0.5%, respectively. The reduction rate of reply messages is much lower than that of query messages owing to the more dropped messages of EASE. The larger number of dropped messages of EASE is discussed in

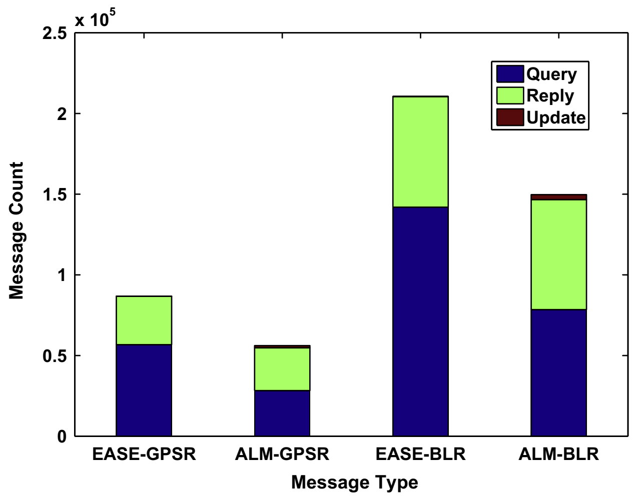

Fig. 16. Message distribution.

Section 4.4. When employing BLR, the multiple paths lead EASE to deliver as many as reply messages of ALM. The splitting and moving storage nodes of ALM result in substantially fewer number of query messages than that of EASE. However, since read-one-write-all policy is used to keep the consistency of all replicated storage nodes, one location update has to incur multiple updates. Thus, ALM produces more update messages than EASE. Fortunately, ALM considers several aspects of transmission messages when making decisions on storage node relocation and replication. Therefore, even though producing more update messages, ALM outperforms EASE in terms of message size.

4.4. Query result distribution

We examine the query results of ALM and EASE in this experiment. The query result received by a querying source is referred to as a success result if the object location information is included in the reply message. Otherwise, the received result without the object location information is called a failure result. For a query, if no reply message is received, it is called a loss result. From Fig. 17, we can see that almost all queries of ALM with GPSR can be answered successfully. On the other hand, EASE with GPSR suffers from a large number of loss results. With the aid of splitting and moving storage nodes, the query and reply messages can be delivered within fewer hops. Since the number of experienced hops becomes less, the probability of message dropping is decreased. As a consequence, ALM achieves higher success result rate. This also explains the poor performance of EASE with GPSR, which has longer paths between storage nodes and querying nodes. With BLR, the multiple paths combat the low success rate problem by increasing the transmission probability of messages. However, the performance of ALM is slightly degraded when using BLR. The reason is that multiple paths causes too many message transmissions in ALM, which has more storage nodes. The large number of message transmissions increases the message collision probability which makes some of the update messages dropped.

4.5. Query response time

In this experiment, we investigate the impact of query rate on query response time of both schemes. The query response time is defined as from the time a querying node issuing a query message to the time the source receiving the reply message. As depicted in Fig. 18, ALM is able to shorten query response time thanks to the relocated and replicated storage nodes closer to querying nodes.

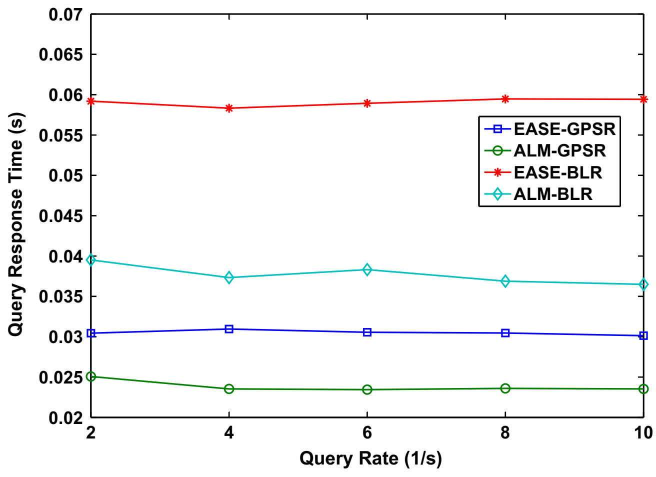

Fig. 18. Impact on query rate vs. query response time.

The improvement ratio of the query response time of ALM over EASE increases from 17.6% to 21.9% as the query rate increases from 2/s to 10/s when GPSR is used as the routing protocol. With BLR, the improvement ratio is between 33.2% and 38.6%. When the query rate is low, the proposed replication and relocation mechanism is rarely executed. Thus, the improvement ratio is not as large as that in higher query rate scenarios. Besides, with BLR, the multiple paths make the query sources receive reply messages more quickly. Consequently, the performance improvement for ALM with BLR is more significant than that with GPSR.

4.6. Energy consumption

We measure the energy consumption of both schemes in this experiment by varying the query rate. The initial energy of each sensor node is set to 5 J. Fig. 19 shows the total energy consumption of the entire sensor network. We can see that ALM consumes less energy than EASE regardless of the query rate because ALM is effective in reducing the number of update and query and reply message transmissions. This result conforms to the experimental result shown in Section 4.2. Specifically, the reduction rates of the energy consumption of ALM over EASE are from 18.2% to 27.4% with GPSR and from 7.1% to 21.5% with BLR. The multiple paths of BLR cause ALM to be less energy-efficient than ALM with GPSR. The second measurement is to observe the minimum resid-

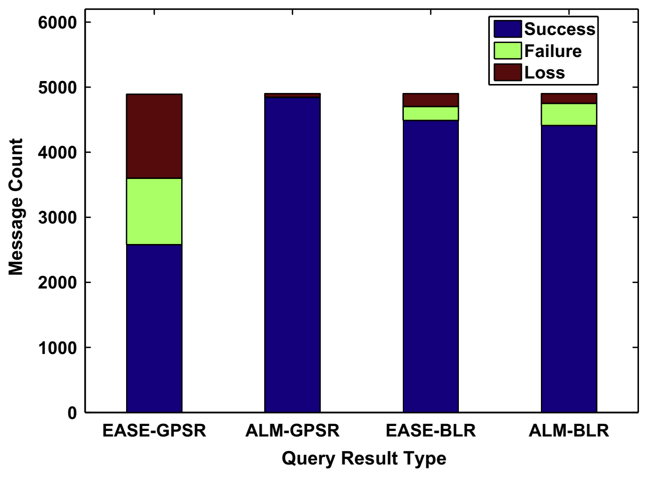

Fig. 17. Query result distribution.

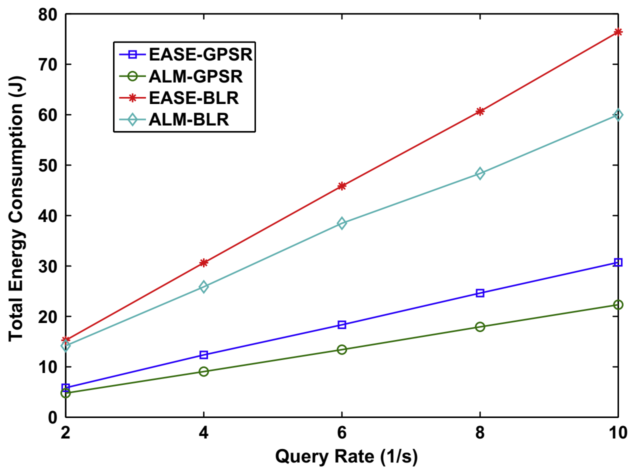

Fig. 19. Impact on query rate vs. total energy consumption.

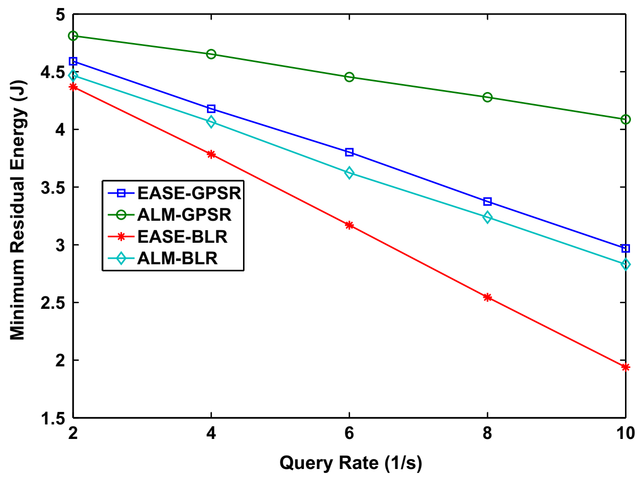

Fig. 20. Impact on query rate vs. minimum residual energy.

ual energy of sensor nodes in the network in Fig. 20. The minimum residual energy of nodes is an indicator of the network lifetime, which is often defined as the time for the first sensor node to run out of its battery. From Fig. 20, we can see that the minimum residual energy decreases in both schemes as the query rate rises. This result agrees with the intuition that higher query rates generate more traffic. In addition, it is observed that the minimum residual energy of EASE drops significantly with the query rate rising. The reason is that the centric storage nodes of EASE cannot move or create replicas and thus the surrounding nodes have to forwarding many messages, thereby consuming their energy quickly. This problem is aggravated by increasing the query rate. On the other hand, the minimum residual energy of ALM decreases slightly as the query rate increases. It is reasonable since the load of sensor nodes is shared after storage nodes invoke the relocation and replication mechanism. Therefore, ALM outperforms EASE in terms of energy consumption and prolongs the network lifetime by distributing the load of sensor nodes.

5. Conclusions

In this paper, we proposed an adaptive location management scheme, called ALM, for approximate location queries in wireless sensor networks. In ALM, we employ a two-tier architecture, which consists of centric and local storage nodes, to facilitate approximate query resolving and reduce the energy consumption of sensor nodes. We also proposed a storage node relocation and replication mechanism, which could adjust the positions of and create or remove replicas of storage nodes according to the location queries and updates. Due to the dynamic position adjustment and replicas of storage nodes, the number of forwarding location update and query messages is reduced by the proposed storage node relocation and replication mechanism, thereby saving more energy. The experimental results showed that compared with EASE [18], ALM is able to reduce the number of transmission messages and the energy consumption of sensor network, and prolong the network lifetime.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38 (4) (2002) 393–422.
[2] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, Wireless Communication and Mobile Computing 2 (5) (2002) 483–502.
[3] C. Gui, P. Mohapatra, Power conservation and quality of surveillance in target tracking sensor networks, in: Proceedings of the 10th ACM International Conference on Mobile Computing and Networking, 2004, pp. 129–143.
[4] Q. Han, S. Mehrotra, N. Venkatasubramanian, Energy efficient data collection in distributed sensor environments, in: Proceedings of IEEE International Conference on Distributed Computing Systems, March 2004.
[5] M. Heissenbuttel, T. Braun, T. Bernoulli, M. Walchli, BLR: beacon-less routing algorithm for mobile ad hoc networks, Computer Communications 27 (2004) 1076–1086.
[6] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, in: Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking, 2000, pp. 56–67.
[7] X. Ji, H. Zha, Sensor positioning in wireless ad hoc sensor networks using multidimensional scaling, in: Proceedings of the 23rd Joint Conference of the IEEE Computer and Communications Societies, 2004.
[8] B. Karp, H. Kung, GPSR: greedy perimeter stateless routing for wireless networks, in: Proceedings of Sixth ACM International Conference on Mobile Computing and Networking, 2000, pp. 243–254.
[9] H.T Kung, D. Vlah, Efficient location tracking using sensor networks, in: Proceedings of IEEE Wireless Communications and Networking Conference, 2003.
[10] J. Li, J. Jannotti, D. De Couto, D.R. Karger, R. Morris, Scalable location service for geographic ad hoc routing, in: Proceedings of the Sixth ACM International Conference on Mobile Computing and Networking, 2000, pp. 120–130.
[11] C.-Y. Lin, W.-C. Peng, Y.-C. Tseng, Efficient in-network moving object tracking in wireless sensor networks, IEEE Transactions on Mobile Computing 5 (8) (2006) 1044–1056.
[12] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, J. Anderson, Wireless sensor networks for habitat monitoring, in: Proceedings of the First ACM international Workshop on Wireless Sensor Networks and Applications, 2002, pp. 88–97.
[13] M.T. Ozsu, P. Valduries, Principles of Distributed Database Systems, Prentice-Hall, Englewood Cliffs, NJ, 1999.
[14] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, S. Shenker, GHT: a geographic hash table for data-centric storage, in: Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications, 2002, pp. 78–87.
[15] S. Tabbane, Location management methods for third generation mobile systems, IEEE Communications Magazine 35 (8) (1997) 72–78.
[16] O. Wolfson, S. Jajodia, Y. Huang, An adaptive data replication algorithm, ACM Transactions on Database Systems 22 (2) (1997) 255–314.
[17] X. Wu, VPDS: virtual home region based distributed position service in mobile ad hoc networks, in: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, 2005, pp. 113–122.
[18] J. Xu, X. Tang, W.-C. Lee, A new storage scheme for approximate location queries in object tracking sensor networks, IEEE Transactions on Parallel and Distributed Systems 19 (2) (2008).
[19] Y. Xue, B. Li, K. Nahrstedt, A scalable location management scheme in mobile ad hoc networks, in: Proceedings of IEEE Conference on Local Computer Networks, 2001.
[20] O. Younis, S. Fahmy, HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks, IEEE Transactions on Mobile Computing 3 (4) (2004) 366–379.
[21] W. Zhang, G. Cao, DCTC: dynamic convoy tree-based collaboration for target tracking in sensor networks, IEEE Transactions on Wireless Communications 3 (5) (2004) 1689–1701.
[22] W. Zhang, G. Cao, Optimizing tree reconfiguration for mobile target tracking in sensor networks, in: Proceedings of the 23rd Joint Conference of the IEEE Computer and Communications Societies, vol. 4, 2004.