國立交通大學

網路工程研究所

碩 士 論 文

在無線網路中估計位置的服務管理

Approximate Location Management in Sensor Networks

研 究 生：吳承恩

指導教授：黃俊龍　教授

中 華 民 國　九 十 六　年 九 月

在無線感測網路中估計位置的服務管理
Approximate Location Management in Sensor Networks

研 究 生：吳承恩　　Student：Cheng-En Wu

指導教授：黃俊龍　　Advisor：Jiun-Long Huang

國 立 交 通 大 學

網 路 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

September 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年九月

在無線感測網路中估計位置的服務管理

學生：吳承恩　　指導教授：黃俊龍

國立交通大學資訊學系網路工程所碩士班

摘　　要

在最近幾年來，物件追蹤的一直是無線感測網路中重要的應用之一。而其中物體位置的服務管理更是受到重視的議題。而良好的物體位置管理系統可以讓無線感測網路中傳輸訊息更有效率進而使感測器更加省電以達到延長使用壽命的目的。因此，我們提出了在無線感測網路中估計位置的服務管理，並且使用動態伺服器來減少整體網路中的訊息流量。在物件追蹤的應用中，有部分的使用者作查詢時，可以容許某種程度上的誤差。而估計位置管理便是針對有誤差的查詢來做出位置管理的服務以減少更新訊息的流量。然而，在過去的位置查詢服務系統中，位置伺服器都是固定在同一個感測點上。因此，我們也提出了一個動態伺服器的方法。此方法會針對訊息目前的流量與趨勢去調整自己的位置來更進一步的節省訊息的數目。最後，我們用模擬程式去探討本方法的的效能。我們的方法不但能減少整體網路的訊息流量，也能減少電量的消耗以及分散負擔給其餘感測點。

Approximate Location Management in Sensor Networks

student：**Cheng-En Wu** Advisors：Dr. **Jiun-Long Huang**

Institute of Network Engineering College of Computer Science
National Chiao Tung University

ABSTRACT

Location management is an important issue of wireless sensor networks in the recent years. The location data storage management scheme of object tracking will affect the energy efficiency of sensor nodes since the transmission is the main factor of power consumption. Therefore, we propose an approximate location management scheme with dynamic servers to reduce the total message count of networks. Since some users can tolerate the imprecision location data, they send query messages with an error bound. Then the location servers can reply an imprecision location data to users and the targets need not to send update messages all the time. So we can utilize the approximate queries to reduce the update message cost. In the previous works, the positions of servers are fixed. Therefore, we also propose the dynamic servers scheme. The dynamic servers are storages which store the location data of targets and adjust their positions according to the message flow. To evaluate the performance of the proposed scheme, several experiments are conducted. The experimental results show that we reduce the message count, save the energy consumption, and balance the load.

誌　　謝

能完成這篇論文，首先得感謝我的指導教授黃俊龍老師。由於老師是剛進學校的新老師，因此必須得花特別多時間與精力在準備課堂以及實驗室整頓上。然而，老師還是很認真地指導我的論文，一直指出我方法上的缺失並且給予許多的建議，也讓我了解到做研究的態度。多虧了老師，我這兩年獲益匪淺。

再來要感謝實驗室的夥伴們，冠翰、信翰以及瑞男。從一開始大家進這個新實驗室就能互相幫忙。每次在做研究遇到瓶頸時，他們總是能提醒我正確的方向。在準備口試時，大家也能互相指導。也要感謝王禾學弟，總是不辭辛勞幫我跑腿。還有欣怡學妹總是會提議大家出遊烤肉，來減輕我們的壓力。國禾學弟雖然一開始很安靜，但熟了會發現是個很幽默的人。建平除了是實驗室夥伴也是室友，因此受到他更多的照顧了。實驗室有你們這群人讓我過的很開心，非常感謝你們。

也感謝我的好朋友們。做研究很煩悶時，總是會陪我打球聊天的言叡、會長、喬吉、馬又、展岱、逼引、大雕以及鳥宇等。還有陪我修課的賀伯、小豆、基妮以及小瞇，每次當我分身乏術時，總是能教導我作業上的困惱。這兩年來真的很感謝你們，你們總是帶我撐過研究生生活中最痛苦的時期。

最後我要感謝一直以來培育我的父母，沒有他們我絕對不會完成這篇論文。每次研究或課業一忙無法回家時，父母總是能體諒我。我真的非常感謝他們對我的用心與包容。

目　　錄

表目錄

圖目錄

Chapter 1

Introduction

A sensor network is wireless network composed by sensor nodes. A sensor node is a cheap device which has sensing, computing and communicating capabilities. It collects environmental data, processes these data and transmits them to another sensor node by wireless ad-hoc network. In data transmission, the sensor node can be a storage storing the data or be a router forwarding the message. By using sensor networks, people can deploy the sensor nodes to collect data in their interested fields.

Energy conservation is always one of the major issues of sensor networks in recent years since sensor nodes have limited power. Since sensor nodes are usually deployed in areas where people are hard to arrive, the batteries of sensor nodes is difficult to replace. Even the sensing field is indoor, replacing the large number of sensor nodes is also time consuming. Therefore, many studies focus on saving energy consumption to prolong the lifetime of sensor nodes.

In sensor networks, there are many useful applications to monitor variations of environment in a fixed field. The most used one is object tracking. Sensor nodes detect a target's existence and store its location information in a fixed storage. When the target moves, sensor nodes are responsible to track its location information for user to query later. There are many applications of object tracking. For example, monitor wild animals or track military tanks [9][17]. These applications involve detecting, positioning, data transmitting in sensor networks and many efficient schemes have been proposed in the recent years [2][3].

Many efficient schemes for object tracking in wireless sensor networks have been proposed in recent years. Theses works can be classified into two classes. The first one is detecting and

positioning. It includes the technology to position the target's location or to design a sleep schedule for sensing nodes to turn into sleep mode when the target is not in their sensing ranges. The second class is location data management. The location information of a target will be put in a specific node for querying and updating. So the challenge of location data management is to let the position of this specific node be known by a target and users who query. Location data management is the process to design querying, updating, and defining the position of a node where the location information is put. In this thesis, we will focus on the second class.

One widely adopted approach in location data management is sink-based storage scheme. The sink is an external node and the position of sink is known for all nodes in sensing field before deployment. In sink-based storage scheme, the responsibility of sink is to store the data updated by sensing node and to reply the answers to querying nodes. An alternative approach is in-network based storage scheme. It means the storage node is in the network field, not in an external node. Data-centric storage (abbreviated as DCS) [10] scheme is the most frequently mentioned in-network based storage scheme. In this scheme, the event of target is classified and named. When the event occurs, it applies a hash function to calculated the position of storage node. For example, we give an unique identity to each target first. When sensing node detects a target, the identity of target is the input of a hash function and the output of the hash function is the address of the storage node. The sink-based scheme is suitable for the cases that query sources are out of sensor network. On the other hand, the in-network based scheme is suitable for the cases that query sources are in the network.

DCS is a centralized storage of in-network based scheme. Besides the centralized storage scheme, local storage scheme [13] is one of recent investigations to conserve energy of sensor networks in object tracking. This scheme utilizes the benefit of approximate query to generate the local storage. It allows user to send location query with an error bound and location data of target stores in hierarchical way. The higher precision data are stored in local storage which is nearest with target and the lower precision data store in centric storage. Thus, most location updates are sent to only local storage when the target moves within approximate radius. Therefore, approximate location management can reduce the amount of location update message. In previous work [10][13], the location server is fixed for each target as shown in Figure 1.1(a). It may cause a lot of message when the target and query users are all far away for location server.

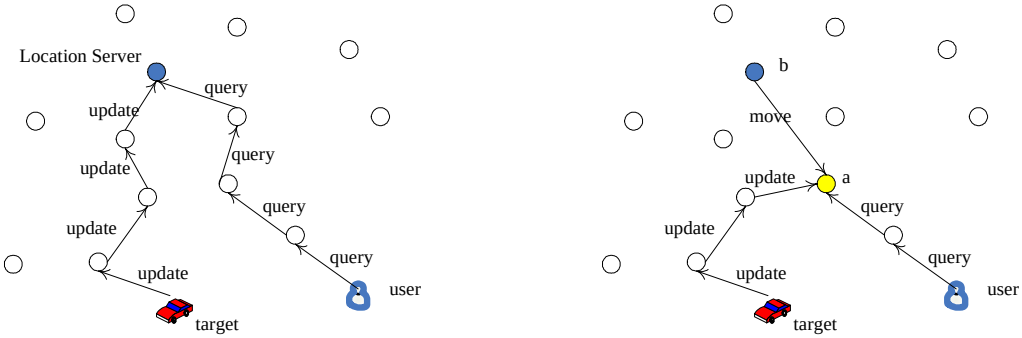


Figure 1.1: The benefit of dynamic server

Both query and update message are routed long distance and the sensor node on the routing path will consume more energy.

To reduce energy consumption in this thesis, we propose an approximate location management (ALM) and develop an adaptive location server scheme to adjust the positions of servers according to the sources of queries and updates. In Figure 1.1(b), we let location server move closely with target and query users when they are all far away from server or create a replica server to serve a hot area which query occurred many times in a short period.

To evaluate the performance of the proposed scheme, several experiments are conducted. The experimental results show that the total message count decreases obviously with high query rate in our scheme. The success rate of queries is always over 90% even the servers are dynamic. The access latency is also decreasing. We also evaluate the energy consumption to show the ALM can balance the load of each node of networks.

The rest of this thesis is organized as follows. Section 2 reviews the related work. The system overview and the details of ALM scheme are presented in Section 3. Besides the snapshot queries, we also handle the continuous queries in our scheme which is described in extension. Then we evaluate the performance of ALM in Section 4. Finally, Section 5 concludes this thesis.

Chapter 2

Related Work

The location management [6][13] consists of two basic processes: querying process and updating process. The target updates its location when it moves and the user queries the interested target's location. When the object move into this filed, the sensor nodes can sense the object existence and report it to server. The server could be a sink which connects to the personal computer or node itself. Then users send query messages to the server to get information of the target. Many useful applications of location management are proposed in the past. Object tracking [5] [8] [13] and location service [6][12][14] are the main applications. In object tracking, people deploy the sensor nodes in the sensor filed which they are interested in. In location service, the ad hoc nodes have to know the location of their destinations when using location-based routing protocol.

The concept of location management is brought up in PCS network in the last [11]. In PCS network, the message count and the latency are the major factors. It uses the two-tier location servers to decrease the update cost [7]. It is similar with some methods proposed in object tracking of sensor networks [12] [13]or location service of ad-hoc networks [14]. To decrease the update cost, they all use hierarchical location server.

In object tracking of sensor networks, sensing and communicating are two energy consuming operations of sensor nodes. So the methods proposed in object tracking are also divided into two categories. One is trying to conserve the sensing power [1][16]. In QoSv[1] and DCTC[16], there are two major operation modes in sensor node, sleeping mode and sensing mode. All sensor nodes in the filed turn to sleeping mode initially. The sensor nodes only wake up and sense the target when the target is around them. They predict the target moving direction to notify each

sensor node that the target will move into its sensing range. The sensor nodes which receive the notify message should wake up and turn to sensing mode. In the other words, only sensor nodes near the object wake up and the others can sleep to conserve the power.

The other category is trying to conserve the communicating power. People in this work believe the transmitting message is main power consumption of sensor nodes especially in object tracking. The target sends the update message when it changes its location and the user sends the query message to obtain the target location. All these two kinds of messages are sent to the server, so the position of server could influence the count of messages in the field. The naive approach is using a fixed external server, sometimes sink, and all messages are routed to this fixed server. This approach is suitable for applications which the queries are from external networks [5][8]. Most of these works focus on decreasing the update message count since the query message has no relationship with sensor nodes. In DAB [5] and DAT [8] , they use hierarchical tree structures to route update messages at some internal nodes in the network field. Then the sink has to forward query messages to those internal nodes to get information. It reduces the update message but produces some query messages. However, when the queries are generated from anywhere in field, both query and update messages have to take into account. Note that the fixed server will produce many query messages. So many methods try to distribute the servers in the field [6][10][12]. Besides distributing the servers, they also organize these servers as a hierarchy. It decreases not only the load of servers but also the amount of messages sent.

In sensor networks, the transmitting and sensing power consumption are two major consideration we mentioned before. But in ad hoc networks, they concern not only the energy consumption but also the reliability since the ad hoc nodes are mobile. In DAT[8] and DAB[5], the nodes of network are organized in tree structures. The GLS[6] cuts the field into grids. Besides of the normal queries, there are some works focus on approximate queries [14]. The data can divided in different accuracy level. The lower accuracy data can be stored at the home server and the higher accuracy data is stored at the local server. This can help some approximate queries get the response at server directly instead of forwarding the query. So the update overload and query overload can be decreased in the works which design for approximate query.

Using approximate query[13] can increase the benefit of hierarchical location server since the location data of each level server has different detailed level data. Sometimes the object moves

with in the acceptable range, then it only updates it's location to local server which is near the object instead of the home server. This action is called local update. Although the local update saves the update message, it makes the query forwarding problem since the server does not have detailed information of the object. The query forwarding problem is server may not have the idea of the answer when the query arrives and has to forward the query to the other low-layer server. Obviously, it will lead to long query latency and the amount of query message in the networks. The trade off in query and update is always studied. Some of works reduce the update overload but increase the query overload and some reduce the query overload but increase the update overload. The comparison of the mentioned related works is give in Table 2.1.

In these previous methods, they all assign a fixed server for each object and distribute them in the field to distribute the load. Beside this fixed server, the sensor or mobile node in the field could be a temporal server. These temporal servers have more detailed location data of object if they are near the object. However, using the fixed server is still not reliable since it produces high overhead and the position of the server significantly influences the message count. Therefore, we propose approximate location management scheme to let server be dynamic to fit the message flow and balance the load.

	Structure	Query Forwarding	Local Update	Data Accuracy Level
DAT[8], DAB[5]	Tree	Yes	Yes	No
GLS[6]	Grid	Yes	No	No
DLM[14]	Grid	Conditional	Yes	Yes
GHT[10], VDPS[12]	None	No	No	No
EASE[13]	None	Conditional	Yes	Yes

Table 2.1: Comparison of related works

Chapter 3

Approximate Location Management

3.1 Overview

Before we introduce the overview of our work, we illustrate the update and query process in basic system first. This basic system of our work refers to EASE[13], an approximate location service by storing imprecise location data at some designated node.

There are two assumptions in our work:

1. Each node knows its location by some device, like GPS.
2. The storage of each node is large enough to store all target's data.

Consider the update process shown in Figure 3.1. There is only one fixed home server for each target. We use the hash function [10] to get the location of home server. A home server is responsible for keeping the imprecise location data of a target and replying this information to user who can tolerate this error bound. A target sends its precise location data to a specific node near the target. We call this node local server. The update process consists of the following steps.

Step 1 A target sends the local update message to the local server periodically. This message includes the id of and the current location information of the target.

Step 2 Local server receives the local update message. It stores the location information of the target first. Then it computes the difference of the target's current location and the

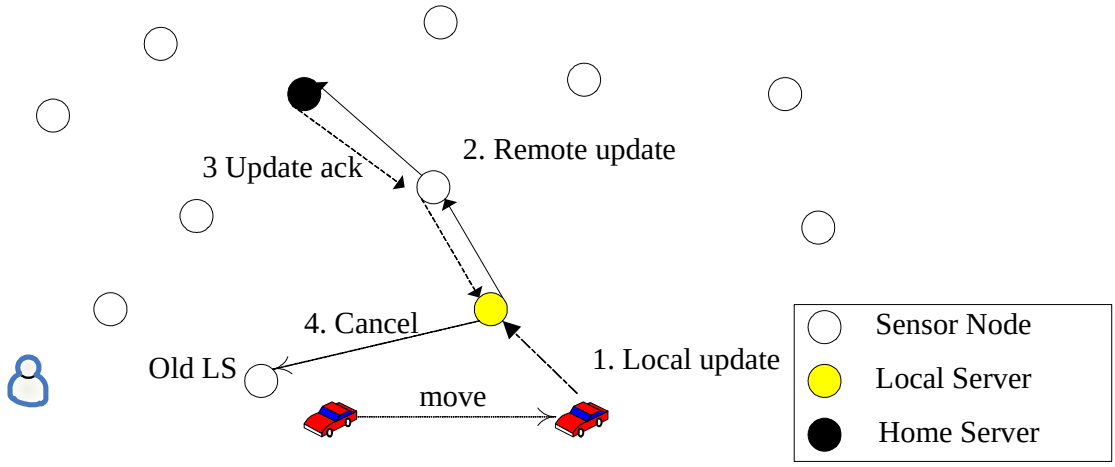


Figure 3.1: Update process

location that last update to home server. If the difference is less than the approximate radius, the update process is terminated. If the difference is larger than the approximate radius, the local server has to send remote update message to home server. This message includes the id of target, the current location of target, the id of local server, and the location of local server.

Step 3 Home server receives the remote update message and store the information of a target and the local server. Then home server replies the update acknowledge message to the local server. If local server does not receive the update acknowledge message within a period, it resends the remote update message to avoid home server miss the message.

Step 4 Local server sends the cancel message to the old local server to delete the record of target.

There are two cases in query process. The first one is the imprecision data in the home server is sufficient to reply the approximate query. Another case is only local server has the precision data. In first case, the home server can directly reply when it receives the query message. As shown in Figure 3.2, the following steps are the query precess in the first case.

Step 1 Querying node sends the query to the home server with target's id and the error bound which user can tolerate.

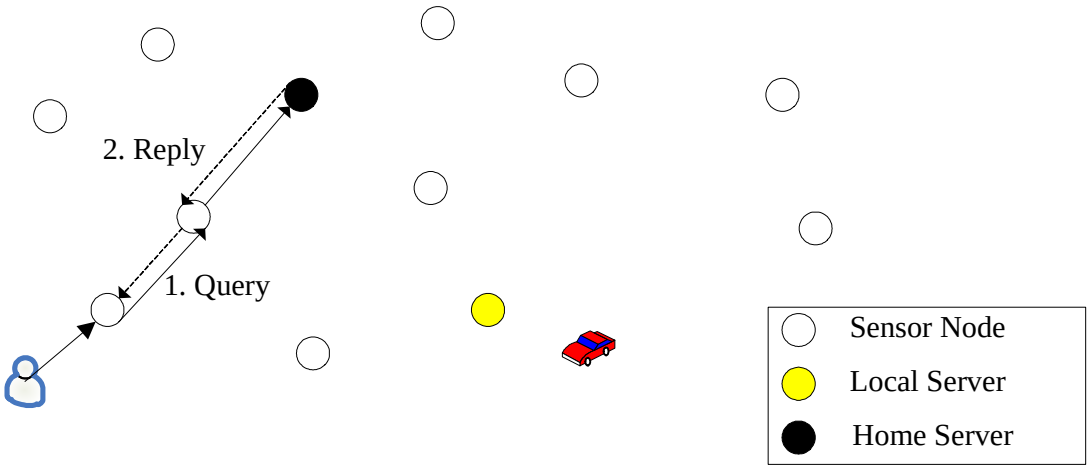


Figure 3.2: First case in query process

Step 2 Home server receives the query message. It compares the error bound of query and the approximate radius of the target. If the error bound is larger, it means the user can tolerate imprecision location information which keeps in the home server. Therefore, the home server replies to the querying node immediately and the query process terminates.

The second case is only local server has the precision data. Then the home server has to forward the query message to the local server. The local server receives the forwarding message and replies to the querying node. The following steps are the query precess in the second case.

Step 1 Querying node sends the query message to the home server with target's id and the error bound which user can tolerate.

Step 2 Home server receives the query message. It compares the error bound of query and the approximate radius of target. If the error bound is less, it means that the user query for the precision data. Then home server forwards the query message to the local server which can reply to the querying node in detail. The forwarding query message includes the identity of target, the identity of querying node, and the location of querying node.

Step 3 Local server receives the forwarding query and replies the answer to the querying node.

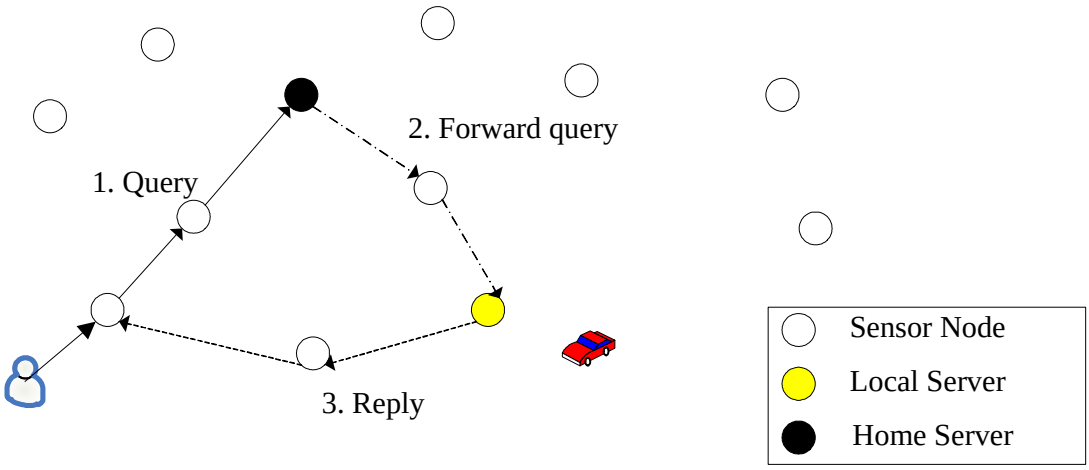


Figure 3.3: Second case in query process

3.2 Dynamic Server

To reduce the count of message transmitted is the main concern in our algorithm. The message count can be decreased if the servers can move closely to users or targets. Therefore, we adjust the positions of servers according to the query and update sources. We will introduce the main operation of dynamic server in the follow two subsection.

3.2.1 Server Moving

When the sources of messages are not distributed uniformly, we should move the position of server to other node to reduce message count. In other words, if the count of messages which come from one of server's neighbors is larger than the sum of other nodes, we change the location of server to the node which has largest flow. We let each server record the message count of its neighbors and check these records every t_{check} period. As shown in Figure 3.4, the location server L receives the total message count T_{n_i} from n_i, a neighbor of L. N is the set of the neighbor nodes of L. The relation of message count to decide the moving decision:

$$T_{n_i} - \sum_{\forall n_j \in N} T_{n_j} \geq \delta_m$$

,where δ_m is the moving threshold.

For example, there are four nodes in Figure 3.5. The node in the middle is the location server

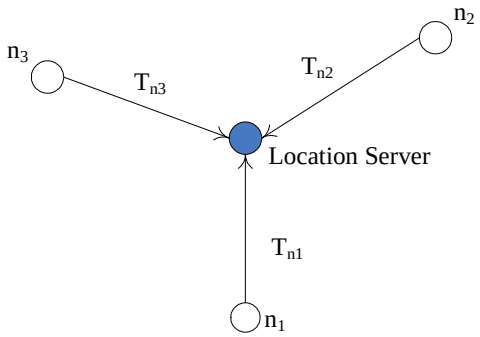


Figure 3.4: Message flow

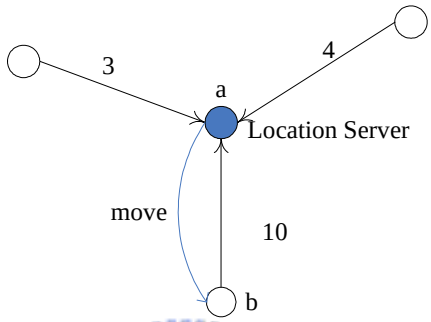


Figure 3.5: Server moving

and it has three neighbors. The number on the edge is the count of message in the period. We can see that the edge between node a and node b has largest flow, and it is larger than the sum of other two nodes. Therefore, we move the location server from node a to node b and set a moving pointer. If the trend of flow in the next period is similar with in the previous period, we could expect the count of message will decrease.

3.2.2 Server Splitting

Besides moving the server, we also split a replica of server when the update rate is lower. Creating more replica will cause redundant update message since the data in the replica has to be consistent. Therefore, we separate the total message into query message and update message to compare the difference between update and query from the neighbor nodes. We find out the neighbor node which has largest count of query message. The server splits a replica to the node if this count is larger than the count of update message. As shown in Figure 3.6, we divide the total message T_{n_i} into query message Q_{n_i} and update message U_{n_i}. The relation of message

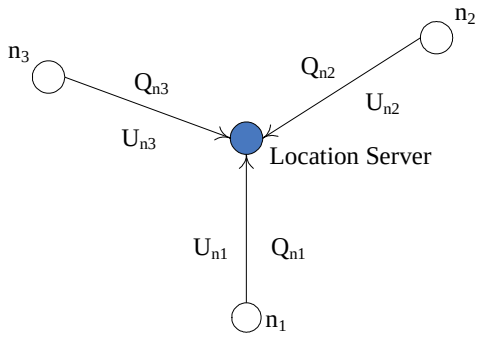
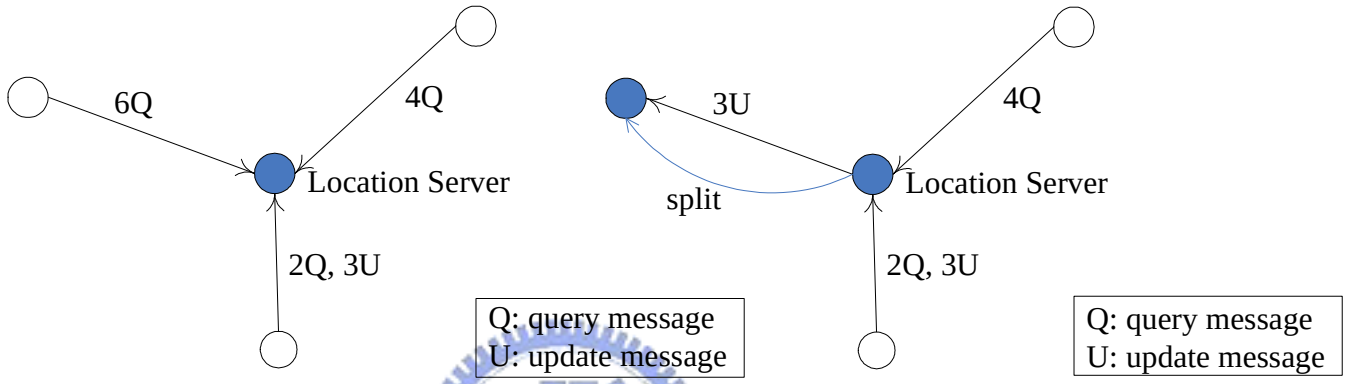


Figure 3.6: Divide the message flow



(a) Before splitting (b) After splitting

Figure 3.7: Server splitting

count to decide the splitting decision:

$$Max_{\forall n_i \in N}\{Q_{n_i}\} - Max_{\forall n_j \in N}\{U_{n_j}\} \geq \delta_s$$

,where δ_s is the splitting threshold.

There is a example in Figure 3.7. Before splitting, the left node has sent the most query message to location server and this count is larger than update message. Then, the server split a replica and a splitting pointer to this node. Even if the location server has to forward the update message to the replica after splitting, the query message will not send to location server again.

3.2.3 Server Merging

The dynamic server split a replica node when query rate is higher than update rate. However, if the update rate is increasing and exceed the query rate at next period, the replica node will

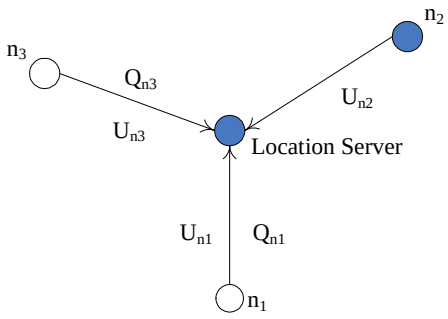
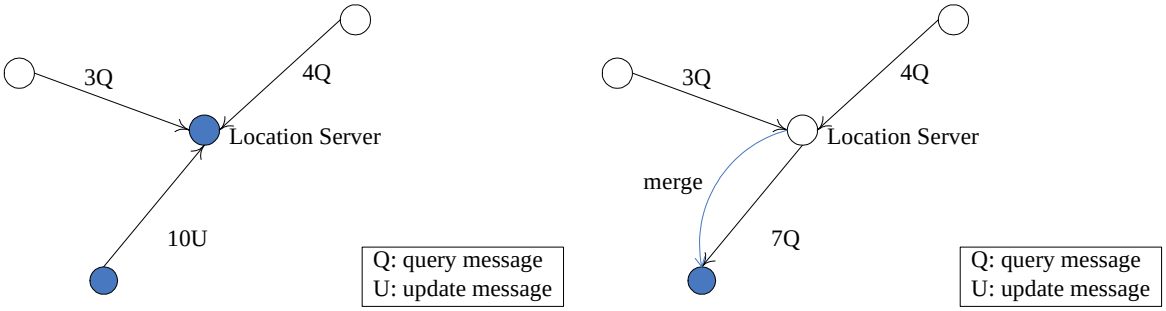


Figure 3.8: Message flow at replica node



(a) Before merging (b) After merging

Figure 3.9: Server merging

be an overhead. Therefore, we merge the replica when the replica node detects the query rate is higher than the update rate. As shown in Figure 3.8, the replica node receives the update message count U_{n_i} from n_i. N is the set of the neighbor nodes of replica. The relation of message count to decide the merging decision:

$$U_{n_i} - \sum_{\forall n_j \in N} Q_{n_j} \geq \delta_{me}$$

,where δ_{me} is the merging threshold.

Give a example of server merging. In Figure 3.9, the replica node is at center and it receives the update message from another dynamic server. We can see that the update message count is larger than the sum of query message count of the other nodes. Therefore, we merge the replica to save the update message and let query message forward to other server nodes.

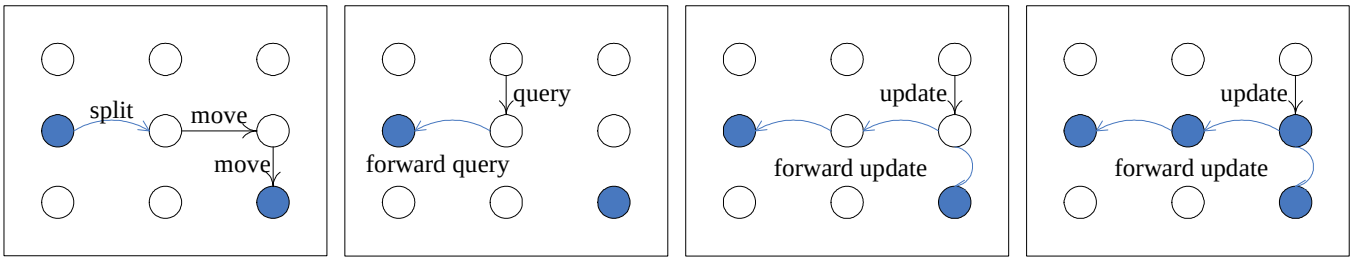


Figure 3.10: Split and move

3.2.4 Discussion

We discuss the possible conditions when we let the server move and split in this section. We consider the situation if a server could move after splitting. There is an example in the most left figure in Figure 3.10. The original server has split once and moved twice, so there are two servers in the network. The second figure shows if the query from the upside node, this query message will be forwarded to the original server or the splitting server. Then the update message has to been routed to these two servers no matter where the update message comes from. We can note that the update message is always routed to the nodes having pointer. If these nodes store the information of target, they do not forward the query message when they receive. Since our goal is reducing the message count, we only allow the splitting server to split and not move anymore.

To avoid ping-pong effect, we cannot let the threshold to be small. However, if we set the bigger threshold, the server moves or splits hardly. We evaluate the performance of threshold setting in section 4. After the server making decision, all the message count record multiplies a ratio α to decrease the effect of history as time passing.

3.3 Query Process

Since we use the dynamic server to reduce the amount of message, the aforementioned update process has to be modified. When the server moves or splits, we set the pointer for routing message to find dynamic servers. The location based routing algorithm, such as GPSR[4], is adapted for routing algorithm of our work. The location based routing always forwards message to node close to destination. Therefore, if the distribution of query or update is inclose in some

area, we can expect the message has to route through some dynamic servers.

Note that the query with lower error bound has to be forwarded to local server in basic system. We add a cache at each node to estimate the approximate radius of target. The local server appends the reply message with a flag to let the querying node know that this information comes from local server. When querying node receives the reply message, it records the error bound of query and stores the location of local server in the cache. At next period, the querying node check this cache first. If the error bound of query is smaller than the estimate approximate radius in the cache, the query will be forwarded to the local server immediately without sending to the home server.

The concept of new query process is searching target's information first when the message arrived at the node. The information of target could exist in each node cause the dynamic server can move or split. In sensor networks, the power consumption of processing is less than transmission. Therefore, the power consumption of searching target's information is relative less and we can omit it. If we find the target's information in the dynamic server before the query message arrived at original home server, the message from dynamic server to original home server can be saved.

The query message includes the identity of target, the error bound which user can tolerate, and the location of query node. At first, the query message is sent to a home server. The node confirms whether itself is the dynamic server of target when it receives the query message. If the query message is routed to home server without meeting any dynamic server, home server has to check if it had moved to another node before. The dynamic server or home server receives the query message and compares the error bound of query and approximate radius of target. If the error bound is larger , the location information of target in home server can reply to the query node immediately. Otherwise, home server forwards the query message to local server to get the information in detail. There is an example in Figure 3.11. The original home server has been split twice so we have three home servers now. The following step is the query process of approximate location management(ALM).

Step 1 Querying node sends the query to home server with target's id and the error bound which user can tolerate.

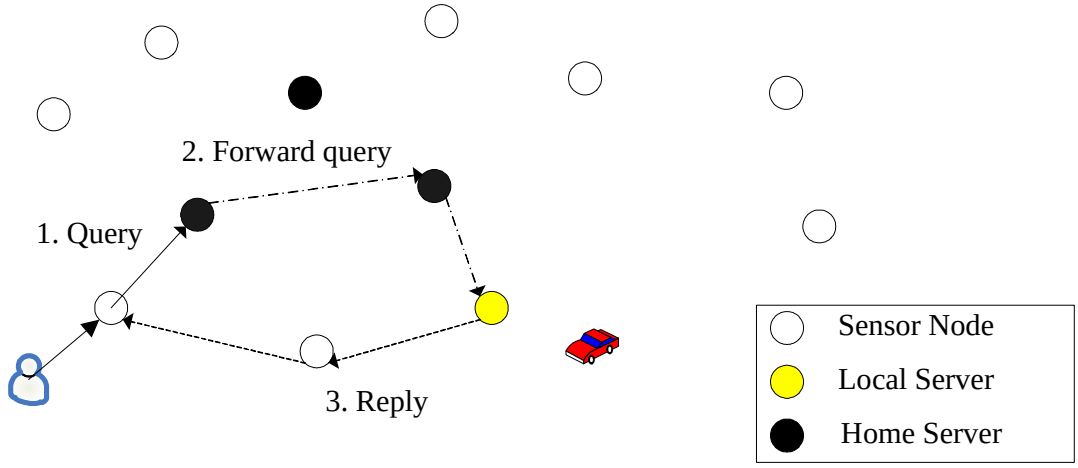


Figure 3.11: First case in ALM query process

Step 2 When the server receives query message, it finds target's information and replies it to the querying node.

Step 3 If the error bound of query does not satisfy the approximate radius of target, the query message is also forwarding to local server.

Step 4 The local server receives the forwarding query and replies the precision data to querying node.

The above situation is first case in ALM query process. The dynamic server can intercept query message and reply by itself without forwarding query to original server. Now, we discuss the second case. The original home server does not have target's information anymore as shown in Figure 3.12. It is possible when we use the dynamic server scheme. The following step is the query process in second case.

Step 1 Querying node sends the query to home server with target's id and the error bound which user can tolerate.

Step 2 When the original home server receives query message, but it does not find any information about target. Then original home server finds the dynamic server by tracing the moving pointer or splitting pointer.

Step 3 Dynamic server receives the query from home server, it finds target's information. Then it replies to querying node directly or forwards the query to local server.

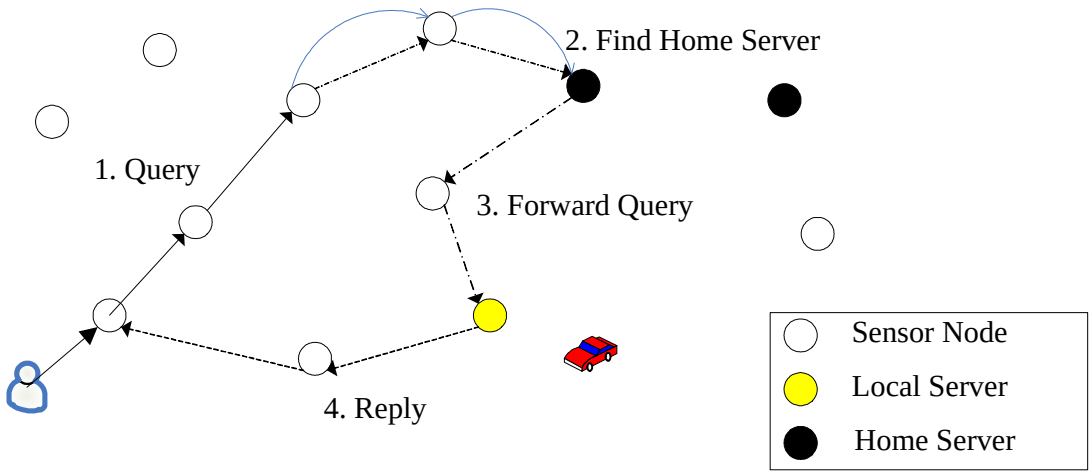


Figure 3.12: Second case in ALM query process

Step 4 The local server receives the forwarding query and replies the precision data to querying node.

Algorithm Protocol Executed at Every Node

```
1:  if receiving a query <target id, error bound> then
2:     if target in home server table then
3:        neighbor table add query count;
4:        if error bound > approximate radius then
5:           reply to querying node;
6:        else
7:           forward query to local server;
8:        end if
9:     else
10:       if is original home server then
11:          find a moving node and forward query;
12:       end if
13:    end if
14: end if
```

3.4 Update Process

The update process is more complicated than the query process of approximate location management since the dynamic server could move to other node or split many times. In Section 3.2, we use the pointer to track the current location of dynamic servers. When the server receives the update message, it not only update target's information but also forward the update message to the node which the pointer set.

3.4.1 Local Update

In this part, we introduce the local update which is similar with EASE[13]. First, the target finds the closest node to be its local server. Then target sends local update to local server at regular interval. When target moves out of approximate radius, it finds the closest node at present to be a new local server and sends local update message. This message includes the location of old local server. When new local server receives the local update of target, it sends cancel message to old local server. After local server receiving the local update, it computes difference between the current location of target and the location that local server updates to home server. If the difference is larger than approximate radius, local server sends the remote update to home server.

3.4.2 Remote Update

Local server sends the remote update to home server when it detects the current location of target is out of approximate radius. The remote update message contains the identity of target, the current location of target, and the location of local server itself. This message will be routed by geographical routing protocol. Once the node receives the message and forwards this message to the node nearest the destination. Now we have to modify the routing protocol slightly to find the dynamic server in our work. When the node receives the remote update message, it does not forward the message first but confirms whether itself is the dynamic server of target. The dynamic servers are the node which the home server moved to or the replicas of the home server. If the node is dynamic server, it updates the information of target. Besides updating, the dynamic server has to check if it has any splitting pointer to other nodes and forward the remote update message to the nodes which the splitting pointers set. If the remote update message does not meet any dynamic server, it will arrive at a home server. Home server updates the information of target if it has no moving pointer. Otherwise, the remote update message has to forward to the node which the moving pointer set. The remote update message terminates when it arrives at a home server or a dynamic server which has no splitting pointer or moving pointer. In Figure 3.13, there are three home servers and two pointers to point split servers. The following steps is the update process in Figure 3.13.

Step 1 A target sends the local update message to local server periodically.

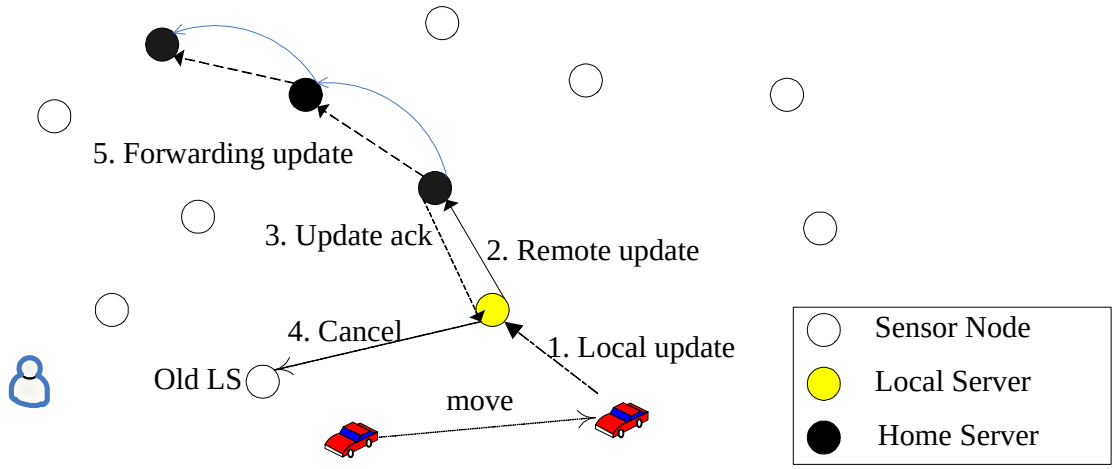


Figure 3.13: Update Process of ALM

Step 2 Local server receives the local update message and sends the remote update message to home server if it is necessary. Home server receives the remote update message and store the information of a target and the local server. Then home server replies the update acknowledge message to the local server.

Step 3 Local server sends the cancel message to old local server to delete the record of target.

Step 4 Forward the update message to other node which pointer set. The nodes receive the forwarding update message and update their data. Then they forward the message to next node.

Algorithm Protocol Executed at Every Node

```
1: if receiving a remote update then
2:    if target in home server table then
3:       neighbor table add update count;
4:       store new location and local server of the object;
5:       find all splitting nodes and forward update message;
6:    else
7:       if is original home server then
8:          find a moving node and forward update;
9:       end if
10:   end if
11: end if
```

3.5 Extension

In section 3, we propose the dynamics servers of location management system in wireless sensor networks. We use the history of message flow to adapt the location of home server and introduce how the snapshot query works. The snapshot query is used to query the location of target at specific time. Since a history of message flow is the main factor of dynamic server, the query distribution of next period is more similar with history will improve the performance of our work. The contrary of snapshot query is continuous query. Continuous query could seen as a fixed query distribution of snapshot queries. A continuous query message has to append the end time and the rate requirement. Users send continuous queries to get the replies from server in a specific rate. This large replies flow encourages us to take continuous query into consideration in our work.

We illustrate how the continuous query works in basic system first. The basic system is a location service without dynamic server mentioned before. We add a new table to deal with continuous query. If home server receives the approximate query and the information in home server can satisfy, it adds an entry in this table and reply the answer in a fixed period. Otherwise, this query message is forwarded to local server and local server will be responsible to reply. Since the local server of a target is not fixed, the hand off problem will be exist. When a target moves for a long time and selects another node to be a new local server, the old local server has to send the entry about target form table to new local server. Therefore, the local update message has to append the old local information to inform new local server. With dynamic server, home server could change its location. So we use same method to solve the hand off problem.

The main modification of our method for continuous query is add more count when node replies. According to the information of continuous query, the server could know the query count in the following period. So we can add the expectable query count directly without receiving a query. However, if we add all count once at one node, the server will only move or split once. The server has to add these count and bring it to next splitting or moving node to add. Therefore, if a user sends a continuous query with a continuous rate r_c and end time t_{end}, the server node adds the count when it replies and the quantity is $t_{check} \times r_c \times t_{end}$ when it checks the neighbor table.

Another extension is load balance. The hot spot problem could cause a small portion of nodes out of energy. In GPSR, each node sends beacon message to maintain the accuracy of neighbor table. We append residual energy to this beacon message to let each node know the residual energy of neighbors. Before a node decides to move or split the server to neighbor, it has to ensure that the energy of neighbor is larger than the threshold which user define. When the node detects that its residual energy is less than the threshold, it sends moving message to the node which has most energy from its neighbor nodes. Certainly, the energy of this neighbor node has to be larger than the threshold.

Chapter 4

Performance Evaluation

4.1 Simulation Model

We developed a simulator based on ns-2(version 2.29)[18] to compare approximate location management with dynamic server scheme to EASE[13]. To use GPSR protocol in our simulation, we also use the routing protocol in HLS(Hierarchical Location Service for Mobile Ad-hoc Networks) patch for ns-2.29[19]. We deploy 100 sensor nodes on a $2000 \times 2000m^2$ field. The field is divided into $200 \times 200m^2$ grid cells. There is one sensor node in the center of each cell. The MAC protocol in our simulator is based on IEEE 802.11 and the transmission range of each node is 250 m. Table 4.1 summarizes the system parameters and setting.

The mobility model in our simulator is linear model. The setting of moving target includes the destination, the start moving time, and the speed. We change the destination every ten seconds to avoid that the target has arrived the destination and be static. The speed of target influences the count of remote update. When target moves faster, the detecting node has to send more remote update message to home server. Therefore, we adjust the speed of target to control the remote update rate. The locations of querying nodes for sending approximate query are issued randomly from the sensor nodes in the field. The error bound of queries are uniformly distributed between 0 and 300 m.

Parameter	Setting
Field Size	$2000 \times 2000m^2$
Number of Nodes	110
Radio Range	250 m
Sensor Sampling Rate	2 sec
Record Reduce Rate (α)	0.5
Checking Period (t_{check})	2 sec
Query Rate	1 - 10 /sec
Error Bound of Query	0 - 300 m
Speed of Target	10 - 50 m/sec
Approximate Radius	100 m
Initial Energy	2J
E_{elec}	50nJ
ϵ_{fs}	$10pJ/bit/m^2$
ϵ_{ap}	$0.0013pJ/bit/m^4$
Threshold Distance(d_0)	75 m
Update Message Payload Size	32 bytes
Reply Message Payload Size	32 bytes
Acknowledge Message Payload Size	16 bytes
Query Start Time	10 sec
Simulation Time	500 sec

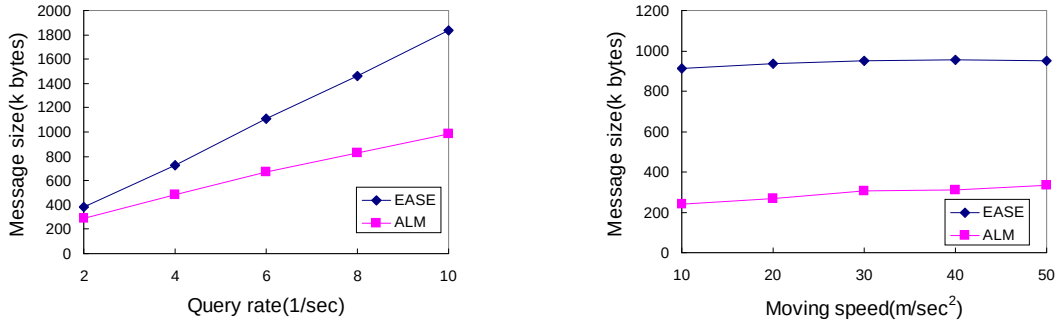
Table 4.1: Simulation parameters

4.2 Simulation Result

4.2.1 Message Size

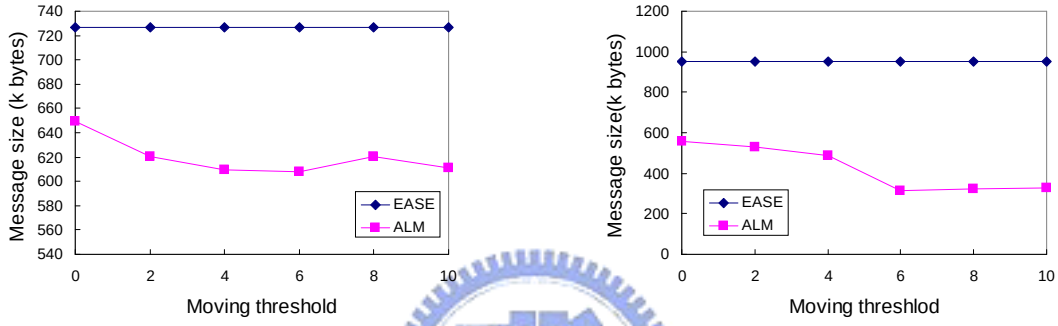
This section evaluates the message size of ALM. We first take the query rate into consideration. Both the merge threshold and split threshold are 3 and the move threshold is set to 6. The speed of target is 30 m/sec. The result is shown in Figure 4.1. The performance of ALM is always better than EASE. The higher query rate can decrease the total message size of ALM obviously and the message size of EASE is in proportion to the query rate. This is because that the lower query rate could waste the move and split message since the benefit from query message is lower. Therefore, our work is suitable for higher query rate.

Figure 4.1 shows the impact on total message size of target's moving speed. The settings of parameter are the same besides moving speed. The query rate is 4/sec. We can see that the moving speed has a little effect on the performance of EASE and ALM. No matter how fast the



(a) Message size vs. query rate (b) Message size vs. moving speed

Figure 4.1: Message size



(a) Random query (b) Bias query

Figure 4.2: Message size vs moving threshold

speed is, the message size of ALM is always less than EASE.

We now evaluate the moving threshold of total message size. At first, we use random query which means the querying sources are randomly from the sensor node in the field as shown in Figure 4.2. To observe the moving situation easily, we set the higher splitting threshold. The splitting and merge threshold are set to 10. The server of EASE does not move or split, so the moving threshold cannot influence the message size. However, the performance of ALM is not stable but it is still better than the EASE. The lower moving threshold causes the server to move more easily. It means that moving server frequently under random query could not get more benefit. The trend of query flow is not apparent when querying source is random. Then we let querying sources are bias. The sensing field is divided into 10×10 grid as we mentioned before. Now, we let all query from the 3×3 grid at lower right corner of field. The result is in Figure 4.2. The performance is the best when the moving threshold is 6. Because the lower threshold

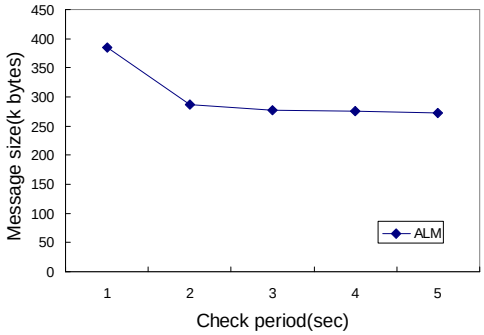


Figure 4.3: Check period vs message size

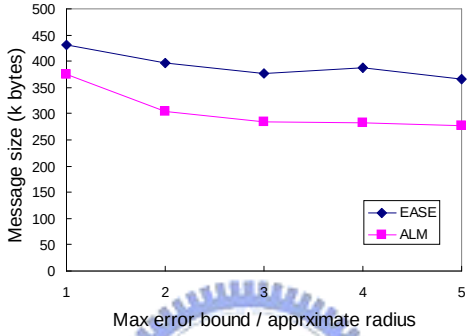


Figure 4.4: Error bound vs message size

makes the ping-pong effect. The server moves into the little grid which the querying sources quickly when moving threshold is lower. Then it moves from one node to another one. After a while, it moves back again and waste the message. However, the higher threshold does not promise the better performance. When the threshold is higher, the server moves slowly and the query message has to route a long distance for a while.

Figure 4.3 shows the performance of ALM to check period. We set the check period from 1 sec to 5 sec. When the check period is 1 sec, the message size is obviously higher. It is because that the check period is too short and the record of location server is not enough to decide if it should move or split.

Both the count of remote update message and forwarding query message are affected by the approximate radius. In Figure 4.4, we compare the performance of ALM and EASE when we set different ratio of error bound and approximate radius. The query rate is 2/sec. We can see the message size of EASE is higher when the ratio is smaller. This is because the query messages

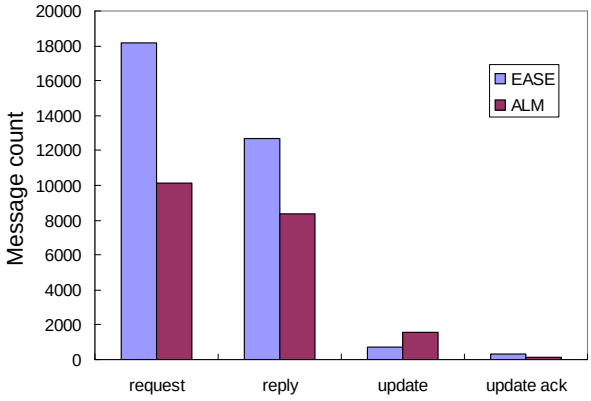


Figure 4.5: Breakdown of message

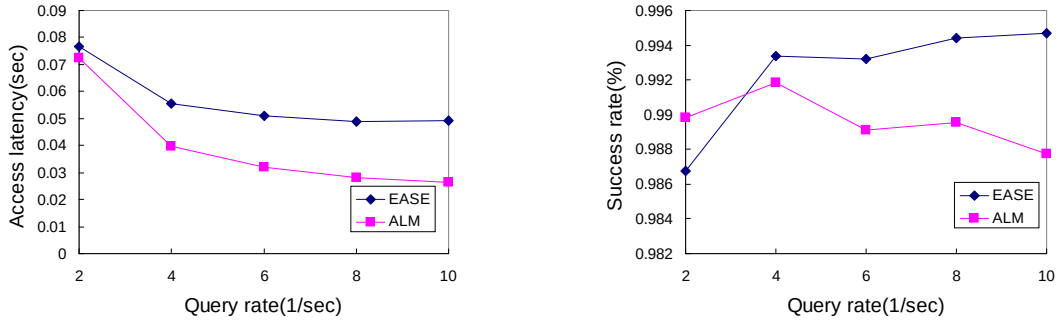
are forwarded when the error bound is smaller than the approximate radius. The smaller ratio makes more query forward to local server and decrease the count of remote update. Therefore, the message size is less when the ratio grows. The slop of ALM is similar to EASE. The reason is the same as mentioned above, but the performance of ALM is better.

To know how ALM improves the performance, we provide a breakdown of message in Figure 4.5. The update message includes the local update and forwarding update. Forwarding update messages are the overhead of replica server nodes. In ALM, it decrease the most part of message in query and reply and incurs a little overhead.

4.2.2 Quality of Query

We also evaluate the quality of query in ALM scheme and EASE scheme. We divide the quality of query into two parts, success rate and latency. The success rate means the percentage of receiving replies. Users could be worry about if they can receive a reply when they query to dynamic servers. To ensure that querying node can receive a reply in ALM scheme, we evaluate the success rate of query rate. The latency includes the total time from sending query to receiving the answer. We expect the latency is lower in ALM scheme since the servers should move or split closely to querying node when query rate is higher. The results are in Figure 4.6.

As shown in Figure 4.6, when the query rate is higher, both the latencies of ALM and EASE are lower. But the performance of ALM is always better than EASE especially in high query rate. The higher query rate encourages the servers to split or move and the routing distance



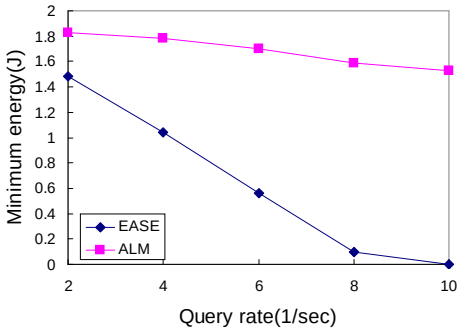
(a) Latency (b) Success rate

Figure 4.6: Quality of query

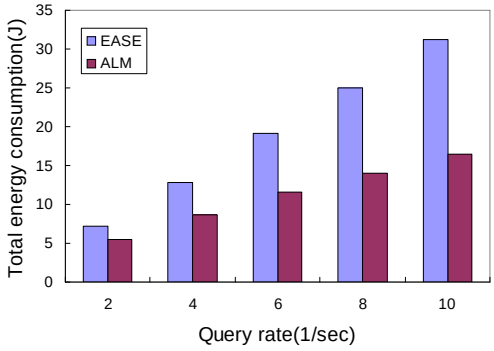
from querying node to dynamic servers is shortened. Then the latency is decreased. The success rate of query represents the reliability of location service. Therefore, we evaluate the success rate of different query rate. The result is shown in right figure of Figure 4.6. Both EASE and ALM has high reliability. The success rate is always over 98%.

4.2.3 Energy Consumption

We now evaluate the energy consumption in this subsection. The energy consumption model and the parameters refer to [15]. First measurement is minimum energy. It means the minimum residual energy of node in the network field. This measurement represents the load balance since the higher minimum residual energy means each node costs less power. As shown in left figure of Figure 4.7, the performance of EASE decreases soon when query rate increases. Because the server of EASE cannot move or split, the nodes surrounding server are routed through a lot of message. In ALM scheme, the performance decrease slightly when query rate increases and the residual energy is more. The splitting servers share responsibility for serving queries, so ALM could balance the load. The second measurement is total energy consumption. We also increase the query rate to observe the effect of energy consumption. The result is shown in the right figure of Figure 4.7. The energy consumption of ALM is always less than EASE since the message count is less that we mentioned before.



(a) Minimum energy vs. query rate



(b) Total energy vs. query rate

Figure 4.7: Energy consumption

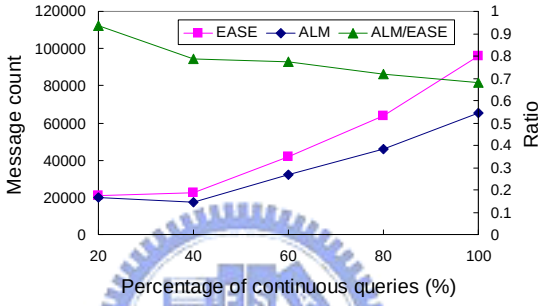


Figure 4.8: Message count vs. percentage of continuous queries

4.2.4 Others

In section 3.5, we illustrate how the continuous query works in our scheme. Now we evaluate its performance in terms of message count and minimum energy to percentage of continuous query. The simulation time is changed to 200 sec. The query rate is set to 1 and each continuous query rate is set randomly distributed between 1 to 4. The continuous query duration is issued randomly between 0 seconds and 100 second. In Figure 4.8(a), with percentage of continuous query is increasing, the ratio of ALM to EASE is decreasing. It means that the improvement of our scheme is better when the amount of continuous queries is more. The load balance is still good in our scheme as show in Figure 4.8(b).

To show the ALM scheme can adapt the message flow, we use two kinds of query distribution to evaluate the message count in each time period. The query distribution is random. The result is shown in Figure 4.9. When simulation just starts, the message count of ALM is the same

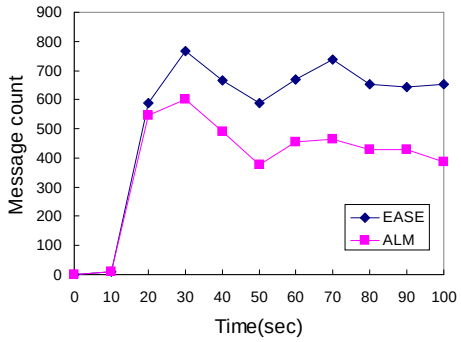


Figure 4.9: Message size vs. time series

as EASE. Then the message count of ALM decreases soon after twenty seconds. It means the dynamic servers adjust their positions according to the message flow and reduce the message count.

Chapter 5

Conclusions

In this thesis, we proposed an approximate location management scheme for object tracking in sensor networks. This scheme consists of approximate queries and dynamic servers. The approximate queries to let target's information be stored in local storage to reduce the update message cost. The dynamic servers utilize the history of message flow to adjust the location of centric storage. We let server move to where has more message and split a replica to serve high query rate region. It reduce total message count of network field and conserve the energy consumption of sensor nodes. The replica nodes also balance the load of servers. We have evaluated the performance of the proposed location management with simulation. The simulation results show that the ALM scheme could improve the performance and outperform the EASE scheme especially in the high query rate.

Bibliography

[1] C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. *In Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, pages 129–143, 2004.

[2] C. Intanagonwiwat, R. Govindan, and 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*, pages 56–67, 2000.

[3] X. Ji and H. Zha. Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling. *In Proceedings of Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 4, 2004.

[4] B. Karp and HT Kung. GPSR: Greedy Perimeter Stateless Routing for wireless networks. *In Proceedings of 6th Annual International Conference on Mobile Computing and Networking*, pages 243–254, 2000.

[5] HT Kung and D. Vlah. Efficient location tracking using sensor networks. *In Proceddings of IEEE Wireless Communications and Networking*, 3, 2003.

[6] J. Li, J. Jannotti, D.S.J. De Couto, D.R. Karger, and R. Morris. Scalable location service for geographic ad hoc routing. *In Proceedings of ACM 6th Annual International Conference on Mobile Computing and Networking*, pages 120–130, 2000.

[7] J. Li, Y. Pan, and X. Jia. Analysis of dynamic location management for PCS networks. *IEEE Transactions on Vehicular Technology*, 51(5):1109–1119, 2002.

[8] C.Y. Lin, W.C. Peng, and Y.C. Tseng. Efficient In-Network Moving Object Tracking in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 5(8):1044–1056, 2006.

[9] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. *In Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, 2002.

[10] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: a geographic hash table for data-centric storage. *In Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 78–87, 2002.

[11] S. Tabbane. Location management methods for third generation mobile systems. *Communications Magazine, IEEE*, 35(8):72–78, 1997.

[12] X. Wu. VPDS: Virtual Home Region Based Distributed Position Service in Mobile Ad Hoc Networks. *In Proceedings of 25th IEEE International Conference on Distributed Computing Systems*, pages 113–122, 2005.

[13] J. Xu, X. Tang, and W.C. Lee. EASE: An energy-efficient innetwork storage scheme for object tracking in sensor networks. *In Proceedings of IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, 2005.

[14] Y. Xue, B. Li, and K. Nahrstedt. A scalable location management scheme in mobile ad-hoc networks. *In Proceedings of IEEE Conference on Local Computer Networks*, 2001.

[15] O. Younis and S. Fahmy. Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach. *In Proceedings of Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 1:629–640, 2004.

[16] W. Zhang and G. Cao. DCTC: dynamic convoy tree-based collaboration for target tracking in sensor networks. *IEEE Transactions on Wireless Communications*, 3(5):1689–1701, 2004.

[17] W. Zhang and G. Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. *In Proceedings of Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 4, 2004.

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

[19] HLS patch for ns-2.29. http://www.cn.uni-duesseldorf.de/staff/kiess/software/hls-ns2-patch.