Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
The Journal of Systems and Software 69 (2004) 159–171
The Journal of Systems and Software
www.elsevier.com/locate/jss

Location management in cellular mobile computing systems with dynamic hierarchical location databases ☆

Guo-Hui Li [a], Kam-Yiu Lam [b,*], Tei-Wei Kuo [c], Shi-Wu Lo [c]

[a] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei, China
[b] Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong
[c] Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, ROC

Received 1 November 2001; received in revised form 1 April 2002; accepted 1 June 2002

Abstract

An important issue in the design of a mobile computing system is how to manage the location information of mobile clients. In the existing commercial cellular mobile computing systems, a two-tier architecture is adopted. However, the two-tier architecture is not scalable. In the literatures, a hierarchical database structure is proposed in which the location information of mobile clients within a cell is managed by the location database responsible for the cell. The location databases of different cells are organized into a tree-like structure to facilitate the search of mobile clients. Although this architecture can distribute the updates and the searching workload amongst the location databases in the system, location update overheads can be very expensive when the mobility of clients is high. In this paper, we study the issues on how to generate location updates under the distance-based method for systems using hierarchical location databases. A cost-based method is proposed for calculating the optimal distance threshold with the objective to minimize the total location management cost. Furthermore, under the existing hierarchical location database scheme, the tree structure of the location databases is static. It cannot adapt to the changes in mobility patterns of mobile clients. This will affect the total location management cost in the system. In the second part of the paper, we present a reorganization strategy to restructure the hierarchical tree of location databases according to the mobility patterns of the clients with the objective to minimize the location management cost. Extensive simulation experiments have been performed to investigate the reorganization strategy when our location update generation method is applied.

Keywords: Mobile computing system; Location management; Location database; Location update cost; Location database reorganization

1. Introduction

Recent advances in mobile communication technology have greatly increased the functionality of mobile information services and have made many novel mobile computing applications a reality. Various innovative applications, such as news updates, real-time traffic information and navigation systems, and real-time stock monitoring systems, are emerging rapidly. One of the most important issues in the design of these mobile computing systems is the location management of mobile clients (Das and Sen, 1999; Pitoura and Samaras, 2001; Plassmann, 1994; Xie et al., 1993). This is not only essential for providing efficient mobile communication services among mobile clients but also important to many new mobile computing applications, such as systems to support *location dependent queries* (Gok and Ulusoy, 2000).

In a cellular mobile network, the whole service area is divided into a collection of inter-connected *cells*. Mobile clients may move within their current cells or move into other cells. While a mobile client is moving, the system has to maintain the real-time locations of its clients. To

☆ This work was partially supported by a grant from the CityU [Project No. 7001259] and National Science Foundation under Grant 60203017.

* Corresponding author. Address: Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong. Tel.: +852-2788-9807; fax: +852-2788-8614.
E-mail addresses: cskylam@cityu.edu.hk, csedchan@cityu.edu.hk (K.-Y. Lam), ktw@csie.ntu.edu.tw (T.-W. Kuo).

0164-1212/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/S0164-1212(03)00082-7

efficiently manage the locations of mobile clients, *location databases* are defined. It is obvious that data items corresponding to the locations of mobile clients are real-time data (Xiong et al., 2001). Their validity may change rapidly with time, especially for the case where the mobility of the clients is high. In order to maintain the validity of the location data items, a mobile client has to generate a location update whenever it moves into a new cell or when it is far from its last reporting location.

In the existing cellular mobile networks, a two-tier location database architecture is adopted to manage the locations of mobile clients (Mouly and Pautet, 1992). In the system, the mobile switching centers (MSCs) are responsible to maintaining the user profiles and their locations. One of the MSCs maintains a location database, called *home location register* (HLR) which keeps the client profiles, including the real-time locations of mobile clients. In addition, other MSCs maintain a *visitor location register* (VLR) for the location information of the mobile clients, which are currently within the cell responsible by the MSC. When a mobile client moves out of its current cell and enters into another cell, a new entry of the client location is added into the VLR of the new cell, and then the HRL will be updated accordingly. In locating a mobile client, the VLR of the cell, where the query is initiated, will be searched first. If the client cannot be found in the VLR, a request will be sent to the HLR of the client to find out its location. Once the cell, where the location of the client is recorded, has been identified, polling messages will be broadcast in the cell to communicate with the client to ensure that it is the right cell where the client is now residing.

This two-tier architecture is simple and easy to implement. However, it has two serious performance problems, which make it not suitable to many new mobile computing applications. Firstly, the number of mobile clients in a mobile computing system can be very large, and the system may need to maintain a large amount of real-time location information for its mobile clients. It is obvious that the two-tier architecture is not scalable. Secondly, because a mobile client is permanently associated with an HLR, the overheads for maintaining the locations of mobile clients can be very heavy if the mobility of clients is high. It is obviously that the cost for locating a mobile client highly depends on the locations of the calling mobile client and the called mobile client. If the called mobile client is far away from the HLR, the cost for locating it could be very expensive.

In order to improve the system performance and to reduce the total cost for locating a mobile client, various strategies have been proposed. One of the efficient methods is to organize the location databases in a hierarchical structure (Pitoura and Samaras, 2001; Pitoura and Fudos, 1998). Although the hierarchical structure of the databases can improve the searching of the locations of mobile clients, the total update overheads for maintaining the real-time locations of mobile clients can also be heavy. In the organization of the hierarchical databases, it is important to consider the update cost as well as the movements of mobile clients, i.e., the mobility patterns. Although various location update generation methods have been proposed in the research of the previous work, most of them are mainly designed for systems using the two-tier location database architecture. These proposed methods may not be suitable to the systems using hierarchical location databases. For example, one of the efficient and widely used update generation methods is the *distance-based method* in which a client will generate a location update when the difference between its current position and its previous reported position is greater than a pre-defined threshold. The number of location databases to be updated under this method depends on how the databases are organized and how the client moves.

At the same time, different mobile clients will have very different mobility patterns, and their mobility patterns may change gradually with time. Therefore, a static hierarchical location database structure may not be able to meet the requirements in minimizing the location management cost. To our best knowledge, it is lack of any detailed study on the relationship between update generation methods and the organization of the location database tree.

In this paper, we study the location update problem in hierarchical databases for mobile computing systems over cellular networks. Our objective is to minimize the total location management cost. The total location management cost consists of two parts: location update (called registration) and locating cost (called paging). A key parameter of the distance-based method is how to define the optimal distance threshold. In this paper, we first design an efficient way to derive an adaptive value for the distance threshold such that the total cost for location management is minimized. Then we propose an effective method to reorganize the location databases to reduce the total location management cost.

The remaining parts of this paper are organized as follows. Section 2 is the related work. Section 3 defines the system model. Section 4 proposes a new method for deriving the distance threshold. Section 5 presents a strategy to reorganize the location databases to reduce the location update cost. Section 6 is the performance studies on the proposed methods. Section 7 is the conclusion and future work.

2. Related work

The research in mobile computing systems has received a lot of interests in recent years. One of the most

important topics is location management. In the past few years, different location update methods have been proposed. Most of them are for systems using the two-tier location database structure. The design of location update generation for hierarchical location databases has received growing interests in recent years. The proposed methods basically can be categorized into the following four basic policies, *location-area*, *time-based*, *distance-based*, and *movement-based*.

In the location-based update method, all of the cells in the system are partitioned into a number of disjointed location areas. A mobile client updates its location when it enters another location area. In the *time-based* method, a mobile client updates its location periodically for every pre-specified time interval (Rose, 1996). In the *distance-based* method, a mobile client updates its location whenever the distance between the current cell and the last registered cell exceeds a pre-defined threshold value. In the *movement-based* method, a mobile client updates its location if the number of cells it has traveled since the last location update exceeds a pre-defined threshold value. In addition to these methods, a paging method (Rose and Yates, 1996) is proposed, where the whole service area is divided into *location areas* (LAs), and the cells in a location area are paged simultaneously. When there is an incoming call to a mobile client, the LAs are sequentially paged for the client following a pre-defined paging strategy. To improve the probability of finding a mobile client and to reduce the paging cost, the system estimates the most possible cell that the client is residing, based on the velocity and the direction information of a mobile client (Wan and Lin, 1999). In Levy and Naor (1999), an active tracking policy (using non-utilized system resources) is proposed to find out the location information of a mobile client. In this paper, we will concentrate on the distance-based update method since it is widely used in the existing cellular mobile networks.

In recent years, a hierarchical location database structure is proposed to organize the location databases in the system (Pitoura and Samaras, 2001). Researchers explored several essential problems, such as location caching, replication, and concurrency control in the hierarchical location databases. In order to reduce the location update cost, some previous works suggested to use location pointers. However, the main concern of this method is that it creates the problem in managing the location pointers. Furthermore, the forwarding pointers can be maintained in any level of the hierarchical location databases, and it is difficult to determine what is the best level to set the forwarding pointers.

Another proposed method to reduce the searching delay and cost is to use caching and data replication to maintain the location information of mobile clients. The main problem of this method is obviously on how to maintain the validity of the cached data items. The overhead can be heavy when the mobility of clients is high. Although using the hierarchical database architecture can speed up the cost in locating a mobile client, to our best knowledge, it is still lack of an efficient location update generation policy for the hierarchical structure. Furthermore, the tree architecture of a hierarchical location databases proposed in the literatures is static and is not suitable to mobile clients with a good variation on the mobility patterns. It is the purpose of this paper to study how to generate location updates based on the distance-based method for systems with hierarchical databases for location management. Our main focus is on the definition of a proper distance threshold for the distance-based method and the presentation of an effective location database reorganization that aims at the reducing of location management cost.

3. Cellular mobile network and location databases model

A cellular mobile network is based on the concept of frequency reuse. The whole service area is divided into a number of connected cells with a base supporting station (BSS) in each cell. The limited wireless bandwidth is partitioned into channels. The mobile clients within a cell communicate with the BSS of the cell through a wireless channel. Since channels are limited resources, how to use the channels is one of the most important issues in mobile network management.

Each BSS maintains a location database. The location databases of different neighboring BSS's are organized into a hierarchical tree structure, as shown in Fig. 1. Note that in an existing GSM mobile network, the location databases are managed by MSCs. It is assumed that there are a large number of mobile clients in the system. They may make a call or submit a query on the locations of other mobile clients. The call distribution is not even in practice. It is assumed that a mobile client usually has a higher probability to make a call to a group of mobile clients than the remaining mobile clients in the system. To simplify the discussion, we assume

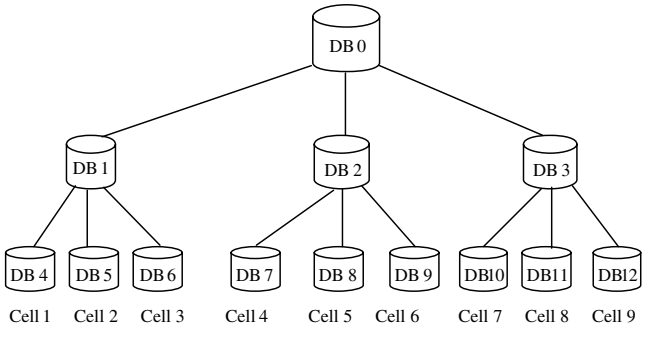

Fig. 1. Hierarchical databases to store the mobile clients' locations.

that each move of a mobile client consists of one to several steps. Each step represents a move across a cell boundary. The mobility of a mobile client is defined by two parameters: (1) the frequency of movement; and (2) the moving distance for each move. It is assumed that different mobile clients have different mobility patterns. Furthermore, the mobility pattern of a mobile client may change with time.

4. Location update generation—a cost-based method

In this section, we will introduce a method for deriving the optimal *distance threshold* for the distance-based update method such that the total location management cost can be minimized. In the distance-based method, when a mobile client x moves from its previously residing cell, $old\_cell(x)$, into the current cell, $cur\_cell(x)$, an update will be generated to update its location to the location database of $cur\_cell(x)$ if the distance between the two cells is greater than a *pre-specified threshold*. (We call it the *distance threshold* in this paper.)

In a hierarchical location database architecture, the organization of the location databases is static, and the system can pre-compute the distance between any two cells and store the distances in a matrix. When a mobile client moves across a cell boundary into another cell, the system can easily query the matrix to get the distance between the old cell and the new cell. However, how to set the distance threshold is not an easy question. Any improper value for the distance threshold could have a serious impact on the system performance, i.e., heavy cost in updating the location of a mobile client or a long delay in locating a mobile client. If the distance threshold value is small, then the locations of mobile clients will be updated very frequently, and a lot of system resources will be consumed on processing the location updates. On the contrary, if the distance threshold value is large, the location uncertainties of mobile clients will be large, and the total cost and time delay for locating a mobile client will become very large.

4.1. Basic principles and definitions

Location management in a cellular mobile computing system basically consists of two procedures: *location update* to report the new location of a mobile client and *paging* for a mobile client. Thus, in the design of location update generation policy, we need to consider the update cost and the paging cost if we want to minimize the total location management cost.

In the cost-based distance threshold method, we first calculate the cost for processing a location update. Then, we compare: (1) the total cost in locating a mobile client when a location update is generated to report the new location of a mobile client; with (2) the total cost in locating a mobile client if a location update is not generated to report its new location. Finally, we calculate the optimal distance threshold value by letting the cost for location update equal to the saving in cost.

An important factor affecting the costs for locating a mobile client and location update is the *call to mobility ratio* (CMR) (Pitoura and Samaras, 2001). It is the average number of calls on a mobile client per cell boundary crossing by the mobile client. If CMR is small, the generation of a location update from a mobile client will not result in much saving in the cost for locating the mobile client. So the generation of a location update may be deferred for this case. On the contrary, if CMR is large, i.e., there are a large number of calls to a mobile client per cell boundary crossing, the generation of a location update can reduce the location cost significantly.

Since the location databases are organized into a tree-like structure, we define the *least common ancestor* of location databases DB_i and DB_j as $LCA(DB_i, DB_j)$. The height of $LCA(DB_i, DB_j)$ to the leaf nodes of the tree is denoted as $lca(DB_i, DB_j)$. (We assume that all of the leaf nodes of the location databases are at the same level in the hierarchical location database tree.) Let the leaf node of a location database for mobile clients at the cell i be $LDB(i)$.

Definition 1. The distance between Cells i and j, termed as $dis(i, j)$, is defined as the height of the least common ancestor of the responsible leaf nodes of the location databases:

$$dis(i, j) = lca(LDB(i), LDB(j))$$
$$\text{If } i = j \text{ then } dis(i, j) = 0.$$

Let $cur\_cell(x)$ be the cell where mobile client x is now residing.

Definition 2. The distance between clients x and y, termed $dis(x, y)$, is defined as the distance of the two cells in which x and y is now residing, namely $dis(x, y) = dis(cur\_cell(x), cur\_cell(y))$.

The above definition captures the locality of two mobile clients. It can be seen easily that a mobile client takes a smaller cost to find the location information of another mobile client if the distance between them is smaller.

4.2. Cost-based distance threshold calculation

In this section, we will derive an optimal solution for generating a location update. Similar to Pitoura and Fudos (1998), when calculating the optimal value for the

distance threshold, we consider the following related costs in location update and the lookup procedure:

F	the cost of sending a message to an arbitrary cell for the case where the system knows at which cell it is now residing
L	the cost of following a link in the tree of the location databases, i.e., sending a message to the parent or the child node of a location database
U	the cost of a database update
Q	a database query cost
P	the cost of polling for a specific client in a cell

To simplify the discussion, we assume that in the hierarchical location database structure, each internal node has d sub-nodes and we denote $dis(cur\_cell(x), old\_cell(x))$ as dis. When a client x moves from cell $old\_cell(x)$ to cell $cur\_cell(x)$, if an location update is generated, the database entries for x in both from $old\_cell(x)$ up to $LCA(cur\_cell(x), old\_cell(x))$ and from $LCA(cur\_cell(x), old\_cell(x))$ down to $cur\_cell(x)$ have to be updated.

To illustrate the searching and update procedure, we can refer to the example hierarchical location databases shown in Fig. 1. It is supposed that mobile client x moves from the cell corresponding to the location database $DB5$ to the cell corresponding to location database $DB8$. To update the location of x, an entry for x's location is added into $DB8$. Then the system searches the location information of x upward in the hierarchical databases until reaching the least common ancestor of $DB5$ and $DB8$, i.e., $DB0$. There is an entry for x's location information at $DB0$. Then, the system continues to search for the location information of x until it reaches the leaf node $DB5$. After that, all the information for x's location in $DB5$ up to $DB0$'s child nodes will be deleted. In each location database from $DB0$ down to $DB8$, an entry for x's location will be added. Thus, the total cost for the location update of x will be

$$\begin{aligned} update\_cost(dis) = {} & 2dis(old\_cell(x), cur\_cell(x)) \times L \\ & + (2dis(old\_cell(x), cur\_cell(x)) \\ & + 1) \times U \end{aligned} \quad (1)$$

It is assumed that after the completion of the location update, a mobile client y in cell $cur\_cell(y)$ calls x. To find the current location of x, a message is sent from the cell in which y is now residing to its parent location database node. If the parent node does not have an entry for x's location, a message is sent to the upper level node until the database $LCA(cur\_cell(y), cur\_cell(x))$ is reached where there exists an entry for x's location. Then, a message is sent downward following the hierarchical database structure until it reaches the current location cell of x. Finally, the system polls for mobile client x.

In the above procedure, we can see that there are totally $2lca(cur\_cell(y), cur\_cell(x))$ times of message transmission and processing, $2lca(cur\_cell(y), cur\_cell(x)) + 1$ times of database queries, and one polling for a specific mobile client in a cell. Thus, the total cost for the case where the mobile client generates an update to report its new location is

$$\begin{aligned} & 2dis(cur\_cell(y), cur\_cell(x)) \times L \\ & + (2dis(cur\_cell(y), cur\_cell(x)) + 1) \times Q + P \end{aligned} \quad (2)$$

If x does not generate a location update after it has crossed a cell boundary, the call from client y will be processed according to x's old location entry in the location databases. After finding that x is not in $old\_cell(x)$, the system polls all the possible cells to find the new location of x. The average number of polling is $1/2 \times d^{dis}$, where $dis = dis(cur\_cell(x), old\_cell(x))$, where a random polling procedure is assumed. At last a forwarding pointer is linked from $old\_cell(x)$ to $cur\_cell(x)$. For example, in the example hierarchical location databases shown in Fig. 1, a mobile client x moves from the cell corresponding to $DB12$ to the cell corresponding to $DB7$ and the related location databases are not updated. When there is a location call for x, the old location information of x will be used first and the system polls for x in the cell corresponding to $DB12$. After finding that x is not in the cell, all the cells which distances from the cell $old\_cell(x)$ are less than the distance between $old\_cell(x)$ and $cur\_cell(x)$ are all the possible cell where x is now residing. They will poll for x until x is found. At most, there are d^{dis} cells to be polled. The average number of cells to be polled for x is thus $1/2 \times d^{dis}$, where a random polling procedure is assumed. In the example architecture shown in Fig. 1, at the worst case, all of the nine cells are possible to be polled for locating x and the average number of polling will be $1/2 \times 3^2$ So, the first location call cost without any location update is

$$\begin{aligned} & 2dis(cur\_cell(y), old\_cell(x)) \times L \\ & + (2dis(cur\_cell(y), old\_cell(x)) + 1) \times Q + P \\ & \times \tfrac{1}{2} d^{dis} + F + U \end{aligned} \quad (3)$$

Note that there is no update on the databases of the cells while the system is locating x. A forwarding pointer to $cur\_cell(x)$ is created inside the database of $old\_cell(x)$ (i.e., the cost of U in Eq. (3)).

After the first location call on x has been resolved, the cost for the following location calls on x for the case where an location update is generated from x to report its new location and for the case where no location update is generated from x will be similar. The only difference is that there will be one more database query and one message searching for the no update generation case.

When we subtract Eq. (2) from Eq. (3), we can get the saving cost for processing the first location call as a result of the generation of a location update:

$$(3)\text{–}(2) = 2(dis(cur\_cell(y), old\_cell(x))dis(cur\_cell(y), cur\_cell(x)))(L+Q) + P\tfrac{1}{2}d^{dis} + F + U - P$$

When the distance between the current locations of y and x is as the same as the distance between y's current location and x's previous location, then the total saving cost including the following $(CMR-1)$ times of location calls (under the no update case) will be increased by $(CMR-1)\times(Q+F)$:

$$\begin{aligned} saving(dis) = {} & 2(dis(cur\_cell(y), old\_cell(x)) \\ & - dis(cur\_cell(y), cur\_cell(x))) \times (L+Q) \\ & + F + U + P \times \tfrac{1}{2}d^{dis} + (CMR-1) \\ & \times (Q+F) - P \end{aligned}$$

When the distance between y and x's current locations is different from the distance between y's current location and x's previous location, the above equation is also valid if the following $(CMR-1)$ times of location calls are from the same cell where y is currently residing. Let us assume that the distance between y and x's current locations is equal to the distance between y's current location and x's previous location, the saving cost can be revised as follows:

$$\begin{aligned} saving(dis) = {} & F + U + P \times \tfrac{1}{2}d^{dis} + (CMR-1) \\ & \times (Q+F) - P \end{aligned} \quad (4)$$

Let $saving(dis) = update\_cost(dis)$, then

$$F + U + P \times \tfrac{1}{2}d^{dis} + (CMR-1)\times(Q+F) - P = (2dis+1)\times U \quad (5)$$

We then calculate the distance value, $Distance\_Threshold_{cost}$, which satisfies Eq. (5). It becomes the *distance threshold* for location update generation. When the distance between the new cell and the old cell of x is more than $Distance\_Threshold_{cost}$, the location update cost is smaller than the saving cost due to the location update. For this case, the mobile client should generate a location update.

One of the advantages of the proposed method for calculating the distance threshold is that it is simple to implement and can be adaptive to the dynamic mobility pattern of the mobile clients. Furthermore, unlike the conventional distance-based method, we do not need to assign a static distance threshold. Instead, the distance threshold is defined based on the system parameters and how a mobile client moves. Every time, a new distance threshold will be defined for a mobile client when it moves across a cell boundary. It can be seen that most of the system parameters can be computed easily before the system setup, i.e., the distance between the cells can be pre-calculated and stored in a symmetric *distance matrix A*.

5. Location database reorganization

5.1. Organization problem

As shown in the previous section, the location update cost is proportional to the distance between the starting and destination cells of the mobile client. In the presented distance-based location update policy, only the movement with a distance larger than the distance threshold will generate a location update. However, we should point out that the distance between two location databases is determined by how the location databases for the cells in the hierarchical location database tree are organized. Suppose that there is a mobile computing system with a geographical topology as shown in Fig. 2 and the location databases be organized as shown in Fig. 1. Note that even if two cells are neighbors to each other, the distance in terms of the location database hierarchy between the two cells may still be significant. For example, Cell 6 is neighboring to Cell 9, but the distance between them is 2 in the hierarchical location database tree.

The hierarchical structure of the location databases can have a significant impact on the location management cost. Before constructing the location database tree, we should consider the mobility patterns of all the mobile clients to better optimize the total location management cost. However, the mobility patterns of mobile clients are not static and may change over time so that the location update cost may only augment gradually. The restructuring of location databases should happen whenever it is necessary to reduce the total location update cost.

For example, referring to the cell organization shown in Fig. 2, if the border-crossing of mobile clients between Cells 3 and 4 is more frequent than that between Cells 2

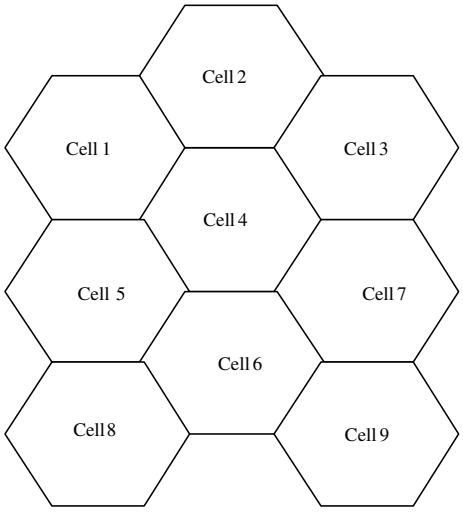

Fig. 2. A mobile computing system consisting of nine cells.

and 3, the system should consider to reorganize the database hierarchy to put the location databases *DB*6 and *DB*7, which are responsible for Cells 3 and 4, in the same cluster in order to reduce the update cost.

Let us consider a more detailed example to illustrate the problem. Suppose that, in the last period, there are 200 times of border-crossings between Cells 3 and 4, and 50 times of border-crossings between Cells 2 and 3. According to Eq. (1) above, if all of the boundary-crossings generate a location update, the total cost to update the 200 times of border-crossings between the Cells 3 and 4 is

$$200(2 \times dis \times L + 2 \times dis \times U + U) = 200(2 \times 2 \times L + 2 \times 2 \times U + U) = 800L + 1000U$$

where *dis* is the distance (see Definition 1) between the two leaf location databases responsible for Cells 3 and 4.

The location update cost to update the 50 times of border-crossings between Cells 2 and 3 is

$$100L + 150U$$

Therefore, the total location update cost is $900L + 1150U$.

If we reorganize the location databases by switching the positions of *DB*5 and *DB*7 with each other in the hierarchical tree, the total cost to update the 200 times of border-crossings between Cells 3 and 4 becomes

$$200(2 \times dis \times L + 2 \times dis \times U + U) = 200(2 \times 1 \times L + 2 \times 1 \times U + U) = 400L + 600U$$

The total cost to update the 50 times of border-crossings between Cells 2 and 3 becomes

$$50(2 \times dis \times L + 2 \times dis \times U + U) = 50(2 \times 2 \times L + 2 \times 2 \times U + U) = 200L + 250U$$

The total location update cost in the reorganized case will be $600L + 850U$ and the location update cost saving is $300L + 300U$. The cost reduction in location updates can be more significant if we adopt the distance-based location update policy proposed in Section 4.

Suppose the distance threshold for generation of location update is 2. Before the location database reorganization, the total cost for the location updates of the 200 times of boundary-crossings between Cells 3 and 4, and 50 times of boundary-crossings between Cells 2 and 3 is $800L + 1000U$. With the use of the distance-based location update method, 50 times of boundary-crossings between Cells 2 and 3 do not generate location database update since the distance is less than the threshold. After the location database reorganization, the total cost for the location updates of the 200 times of boundary-crossings between Cells 3 and 4 and 50 times of boundary-crossings between Cells 2 and 3 will be

$$50(2 \times dis \times L + 2 \times dis \times U + U) = 50(2 \times 2 \times L + 2 \times 2 \times U + U) = 200L + 250U$$

We can see that after location database reorganization, the total cost for location database updates decreases by 75%.

5.2. Reorganizing location database structure

In this section, we propose a location database reorganization strategy with the objective to reduce the total update cost based on the mobility patterns of clients. The main idea is to maintain a mobility matrix, A, to count the number of border-crossings between any two cells to track the changes in the mobility patterns of clients. Whenever a mobile client moves between Cells i and j, A_{ij} in the matrix is increased by one. When the system detects that the total location management cost is too high, the location databases are restructured based on the mobility matrix such that the cells with a large number of border-crossings will be put in the same sub-tree in the hierarchical location database tree. In this way, the cost for location database updates can be reduced.

5.2.1. The procedure

The system monitors the total update cost. Whenever some value of the mobility matrix becomes very high, i.e., being greater than the threshold, the reorganization procedure will be invoked. The reorganization procedure consists of three steps:

(1) mobility matrix transformation;
(2) location database clustering; and
(3) re-organization of location databases.

5.2.1.1. Mobility matrix transformation. Suppose the location databases are organized, as shown in Fig. 1 and Table 1 shows the mobility matrix for the location databases. From the mobility matrix, we can see that mobile clients move more frequently among Cells 4, 5 and 7, than other cells. However, the two location databases responsible for Cells 4 and 5 form a cluster with the location database responsible for Cell 6. Thus the total location management cost is not optimal.

The mobility matrix of a location database system, as shown in Table 1, consists of nine cells. Each entry in the matrix indicates the number of bounding crossings between the corresponding two cells. For example, the value 100 of the entry $C_{1,2}$ indicates that there are 100 boundary-crossings between Cells 1 and 2. Note that there are more crossings between Cells 1 and 2 than those between Cells 1 and 5 (which corresponds to $C_{1,5}$ with a value equal to 20). Based on the number of boundary-crossings between cells, we cluster the location databases in the location database tree to minimize

Table 1
An example of the mobility matrix for the location database structure shown in Fig. 1

	C1	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*	*C8*	*C9*
C1	*C*	100	0	30	25	0	0	0	0
C2	100	*C*	80	40	20	30	0	0	0
C3	0	80	*C*	0	10	30	0	0	0
C4	30	40	0	*C*	120	0	80	25	0
C5	25	20	10	120	*C*	15	160	30	50
C6	0	30	30	0	15	*C*	0	100	60
C7	0	0	0	80	160	0	*C*	40	0
C8	0	0	0	25	30	100	40	*C*	75
C9	0	0	0	0	50	60	0	75	*C*

the total location update cost. The cells which have more boundary-crossings are clustered in the same subtree in the location database hierarchy.

To determine what is the best way to group cells, we could use the following equation to calculate the *mobility affinity of cells* (MAC):

$$MAC = \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij}(A_{i,j-1} + A_{i,j+1} + A_{i-1,j} + A_{i+1,j})$$

where $A_{0,j} = A_{i,0} = A_{N+1,j} = A_{i,N+1} = 0$.

Basically, the equation calculates the boundary-crossing relationships of a cell with its neighboring cells. We exchange the rows and columns in the matrix to calculate *MAC* for different arrangements of cells. For a mobility matrix with N cells (N-leaf location databases), there are $N!$ different transformed matrixes, and each matrix has a *MAC* value. We choose the matrix with the greatest value of *MAC* for clustering.

Following the above idea, the matrix in Table 1 is converted into Table 2, as shown below, where the transformed mobility matrix has the greatest value of *MAC*.

5.2.1.2. Location database clustering. After the transformation, we partition the mobility matrix into two parts: the upper-left part LU and lower-right part RD. LU includes as many cells as possible and, at the same time, the total size of all the location databases responsible for the cells in LU has to be less than a pre-defined database size threshold. (As it can be seen from the hierarchical structure of the location databases, a location database includes the location information for all of its child location databases. So an inner location database has its capability limitation. The threshold is determined by the capability limitation of a direct parent location database of the leaf nodes.) All the databases responsible for the cells in LU are clustered together. If the size of the databases responsible for the cells in RD is larger than the pre-defined database size threshold, the location databases responsible for the cells in RD is split and congregated in the same way as the above procedure until all the location database sets are smaller than the pre-defined threshold value.

For example, the transformed matrix is partitioned into two parts, as shown in Table 3. The upper-left part is denoted as LU, and the lower-right part is denoted as RD. Suppose the total size of the three location databases responsible for cells $C1$, $C2$, and $C3$ is smaller than the pre-defined database size threshold, then the three databases can congregate together. Suppose that the total size of the six location databases responsible for cells $C4$, $C5$, $C6$, $C7$, $C8$, and $C9$ is greater than the threshold, we partition the six cells into two parts: One includes cells $C4$, $C5$, and $C7$, and the other includes cells $C8$, $C9$, and $C6$.

An alternative approach for location database clustering is to adopt a graph model. We could construct a weighted graph of N nodes, which corresponds to the N-leaf location databases (cells) in the system. The weight associated with an edge between the two leaf location databases (cells), say LDB_i and LDB_j, is the number of boundary-crossings between the two cells. It is similar to

Table 2
The transformed mobility matrix

	C1	*C2*	*C3*	*C4*	*C5*	*C7*	*C8*	*C9*	*C6*
C1	*C*	100	0	30	20	0	0	0	0
C2	100	*C*	80	40	30	0	0	0	30
C3	0	80	*C*	0	10	0	0	0	30
C4	30	40	0	*C*	120	80	25	0	0
C5	20	20	10	120	*C*	160	30	50	15
C7	0	0	0	80	160	*C*	40	0	0
C8	0	0	0	25	30	40	*C*	75	100
C9	0	0	0	0	50	0	75	*C*	60
C6	0	30	30	0	15	0	100	60	*C*

Table 3
Partition of the transformed matrix into two parts

	C1	C2	C3	C4	C5	C7	C8	C9	C6
C1	C	100	0	30	20	0	0	0	0
C2	100	C	80	40	30	0	0	0	30
C3	0	80	C	0	10	0	0	0	30
C4	30	40	0	C	120	80	25	0	0
C5	20	20	10	120	C	160	30	50	15
C7	0	0	0	80	160	C	40	0	0
C8	0	0	0	25	30	40	C	75	100
C9	0	0	0	0	50	0	75	C	60
C6	0	30	30	0	15	0	100	60	C

A_{ij} in the mobility matrix method. As the approach adopted in the matrix method, we could partition the graph into two parts: G_1 and G_2 to minimize the value of $NBCT$, where

$$NBCT = \sum\{W_{m,n} | LDB_m \in G_1 \wedge LDB_n \in G_2\}$$

$W_{m,n}$ is the weight associated with the edge between the nodes LDB_m and LDB_n.

5.2.1.3. Location databases reorganization. The location database reorganization procedure is started up when the ratio of the total distance of the boundary-crossings over a period of time is greater than a pre-defined threshold. We construct a function f as follows:

$$f(t) = \frac{ND(t)}{\text{length of } t}$$

where $ND(t)$ is the total distance of the boundary-crossings which generate location update during time period t.

It is assumed that the function $f(t)$ has a nearly ideal value $f(1)$ for the first unit time when the location database tree is constructed. As time goes by, the mobile clients' mobility patterns may have changed, and the location update cost could become very high. The function value becomes much greater than $f(1)$. At the ith unit time after the location database tree is restructured, if $f(i) \geqslant d \times f(1)$ holds, then the location database restructuring procedure is started up. d is adjusted by taking into consideration of the cost for the restructuring procedure.

The steps of location-database reorganization is as follows:

1. Calculate the mobility matrix M for a time interval.
2. Transform the matrix such that MAC with the largest value.
3. Partition the matrix into two parts F and L. The volume of the location databases responsible for the cells in F is less than a pre-specified size threshold $SV_{\text{threshold}}$. The location databases responsible for the cells in F form a cluster.
4. If the volume of location databases responsible for the cells in L is larger than
   $SV_{\text{threshold}}$,
   Then
   {
   $M = L$;
   GOTO step 2;
   }
   Else

The location databases responsible for the cells in L form a cluster.

5.2.1.4. Matrix distribution and computation cost. When we use the mobility matrix to reorganize the location database structure, we need to resolve two practical issues: where to put the mobility matrix and how to minimize the complexity in deriving the optimal organization based on the mobility matrix. The mobility matrix records the number of border-crossings for each pair of adjacent cells. It receives border-crossing counts from the base stations (which generate a count when a client crosses a cell border). It is obvious that if a single mobility matrix is used and deployed in a fixed location (a centralized scheme), a large number of border-crossing counts will be generated, and the total network communication overhead will become very high.

To minimize the communication overhead for the management of the mobility matrix, an adaptive distributed method is proposed in this paper, in which A_{ij} is placed at the location database $LCA(LDB(i), LDB(j))$. Incrementing A_{ij} by 1 only requires $dis(LDB(i), LDB(j))$ times of wired communication in traversing the database hierarchy. In this way, if a location update is generated when a mobile client x moves from cell i to cell j, then there is no extra wired communication at all for the matrix update.

When restructuring the location databases, we need to gather all the information about the mobility matrix from all the "non-leaf" location databases. The execution frequency for restructuring of the location databases should be low so that the total communication cost will not have too much negative impact on the

system performance. However, if the number of cells in the system is large, the total overhead for finding the optimal solution can still be very high. To minimize the computation overhead, we have design two approximate solutions:

Suppose that there are N-leaf location databases in the system. There are $N!$ different combinations for all the leaf location databases. To get the optimal location database organization, according to the algorithm proposed in this paper, the cost of calculation can be very high. In here, we propose two algorithms to achieve the sub-optimal location database organization. Under the first algorithm, the leaf location databases are partitioned into m groups. In each group, there are n location databases. For each group of n location databases, the proposed location database reorganization algorithm in this paper is applied to get the optimal organization for these n location databases. For all of the N $(N = m \times n)$ location databases, the reorganized structure is sub-optimal. We call this method as *partion-based location database reorganization* (PABR).

For the second algorithm, we count the number of boundary-crossings between each pair of adjacent cells and then calculate the mean value. For any pair of leaf location databases, if the number of boundary-crossings between the two corresponding cells is larger than the calculated mean value, the location databases are reorganized to reduce the distance of the two leaf location databases to 1. By restructuring the location database tree, the total location database entries updated can be reduced. We call this method as *boundary-crossing number comparison-based reorganization* (BOCNUCOR).

5.3. An example

Referring to the location databases shown in Fig. 1 and the system cell organization shown in Fig. 2, the cost for a location update for a mobile client x crossing a cell boundary is

$$2dis(old\_cell(x), cur\_cell(x)) \times L + (2dis(old\_cell(x), cur\_cell(x)) + 1) \times U$$

After reorganizing the location databases, the positions of the nine location databases are the same as the positions in the old tree architecture except that the location databases responsible for cells $C6$ and $C7$ have been exchanged with each other. Therefore, the new location update cost is the same except those for boundary-crossings involving cells $C6$ or $C7$ (as the starting cell or the destination cell).

The location update cost for the boundary-crossings between cell i and cell j is denoted as $COST_{ij}$. For the location database architectures before and after the reorganization, only the $COST_{ij}$ $(i = 4, 5, 8, 9;\ j = 6, 7)$ is different. In the old location database architecture, $COST_{ij}$ are

$COST_{46} = 0$	$COST_{87} = 80L + 120U$
$COST_{56} = 30L + 45U$	$COST_{86} = 400L + 500U$
$COST_{47} = 320L + 400U$	$COST_{96} = 240L + 300U$
$COST_{57} = 640L + 800U$	$COST_{97} = 0$

The above values are obtained by using Eq. (1). For example, when calculating $COST_{96}$ the distance between Cells 9 and 6 is 2 in hierarchical structure in Fig. 1. There are 60 times of border-crossings between Cells 9 and 6 as shown in the matrix. The total cost for updating the 60 times of border-crossings is as follows:

$$\begin{aligned} COST_{96} &= (2 \times dis \times L + (2 \times dis + 1) \times U)60 \\ &= (2 \times 2 \times L + (2 \times 2 + 1) \times U)60 \\ &= 240L + 300U \end{aligned}$$

The total location update cost for these boundary-crossings is $1710L + 2165U$. In the reorganized architecture, $COST_{ij}$ are

$COST_{46} = 0$	$COST_{87} = 160L + 200U$
$COST_{56} = 60L + 75U$	$COST_{86} = 200L + 300U$
$COST_{47} = 160L + 240U$	$COST_{96} = 120L + 180U$
$COST_{57} = 320L + 480U$	$COST_{97} = 0$

The total location update cost for these boundary-crossings is $1020L + 1475U$.

Obviously, after reorganizing the location database, the total location update cost is reduced dramatically. This location update cost reduction will be more obvious if the mobile clients' mobility patterns among the cells change more significantly. If the presented distance-based location update generation policy is adopted, the location update cost reduction could be more obvious. For example, if the calculated distance threshold is 2, before the location database reorganization, the costs are

$COST_{46} = 0$	$COST_{87} = 0$
$COST_{56} = 0$	$COST_{86} = 400L + 500U$
$COST_{47} = 320L + 400U$	$COST_{96} = 240L + 300U$
$COST_{57} = 640L + 800U$	$COST_{97} = 0$

The total location update cost for these boundary-crossings is $1600L + 2000U$. After the location database reorganization, the costs are

$COST_{46} = 0$	$COST_{87} = 160L + 200U$
$COST_{56} = 60L + 75U$	$COST_{86} = 0$
$COST_{47} = 0$	$COST_{96} = 0$
$COST_{57} = 0$	$COST_{97} = 0$

The total location update cost for these boundary-crossings after location database reorganization is

$220L + 275U$. It is only 15% of the cost before location database reorganization.

6. Performance evaluation

We have developed an event-driven simulator to study the performance of the proposed location update generation policy (denoted as ADT in the figures) and the two methods (PABR and BOCNUCOR) for deriving solutions for reorganization. We also developed a simulation program using the distance-based method with a static distance threshold (denoted as SDT in the figures) for comparison. The simulation programs are implemented in CSIM-18, which is a simulation language for the C programming language.

6.1. Simulating system model, parameters setting and performance metrics

In the simulation model, the whole service area is partitioned into a set of neighboring cells. A location database is defined for each cell to manage the locations of the mobile clients within the cell. All location databases are organized in a hierarchical tree structure such that the location databases for the cells are leaf nodes in the location database tree. In our baseline setting, each inner node has 3 child nodes, and 200 mobile clients roam around in the whole service area. When there is a call to a mobile client, the related location databases are queried to determine the position of the callee. Then the callee is paged in the cell recorded in the location database. Under the distance-based location update policy, the distance threshold is set as 1. That is, whenever a mobile client crosses a cell boundary, a location update will be generated, and the location entries for the mobile client in all the related location databases will be updated accordingly. The *call to mobility ratio* (CMR) of all the mobile clients follows a normal distribution. In the experiments, it is assumed that the CMR's of all the mobile units follow an exponential distribution.

The following table summarizes the parameters and the baseline setting:

Parameters	Baseline value
Total number of mobile clients	200
Total number of cells	81
Height of the location database tree	5
Node out-degree for an inner location database	3
Mean value of the CMR for mobile clients	1
Deviation of the CMR for mobile clients	0.5

6.2. Experiment results

Figs. 3 and 4 show the location update cost and total location management cost under the ADT and SDT methods when different mean values of CMR are used. When CMR was smaller, the number of calls to the number of cell boundary crossings would be larger. It is because the mobile clients moved less frequently when CMR was smaller. Therefore, as shown in Figs. 3 and 4, the location update cost and the location management cost decreased as increasing of CMR.

It can be observed from Fig. 3 that the location update cost under ADT was much smaller than SDT, especially when the CMR value was small. The better performance of ADT was due to a smaller number of location updates. Under ADT, some of the cell boundary crossings did not generate any location update. Therefore, the total number of location updates generated was less than that in SDT.

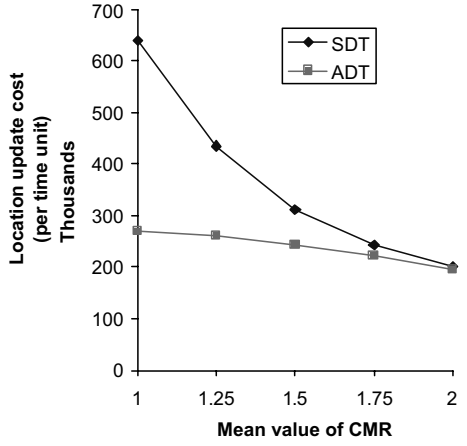

Fig. 3. Location update cost.

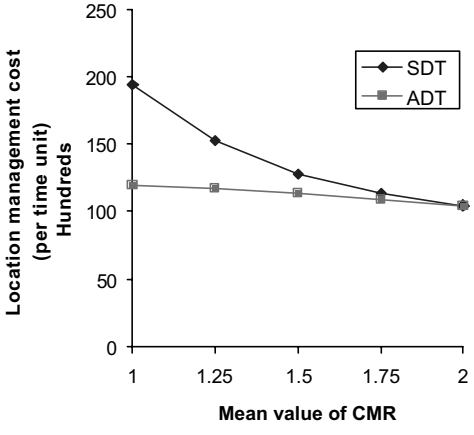

Fig. 4. Total location management cost.

When the mean value of the CMR changed from 1 to 2, the differences in update cost between ADT and SDT became smaller. The reason is that under ADT, if the value of CMR was large, then the distance threshold value would become small. Therefore, each mobile client would generate more location updates on average. When the CMR value was very large, the distance threshold reduced to 1, and all the cell boundary crossings would generate location database updates. For this case, the number of location updates generation under ADT was as the same as that under SDT.

Consistent with the results shown in Fig. 3, the total location management cost under ADT was lower than SDT for different values of CMR. Note that the performance difference between ADT and SDT, as shown in Fig. 4 was smaller, compared with the results shown in Fig. 3 (especially when the value of CMR was large). It was because the total management cost consisted of both location update cost and paging cost. Under ADT if a cell boundary crossing did not generate any location database update, then the system might require to search for the callee. This incurred a higher overhead for locating a client.

Fig. 5 shows the location update cost when the PABR and BOCNUCOR methods were used for location databases reorganization as compared with the case without reorganization (static LDB). In this set of experiments, we used SDT for generating location updates. As shown in Fig. 5, the location update costs using the reorganization methods BOCNUCOR and PABR were consistently lower than the case without reorganization (static LDB). The improvement was greater when a smaller CMR was used. When a larger CMR value was used, the mobile clients in the system moved less frequently. Therefore, the total location management cost would be smaller, and the saving in location update cost would be smaller in BOCNUCOR and PABR. The performance of the two methods PABR and BOCNUCOR was similar, as shown in Fig. 5. Although they used different methods for deriving good

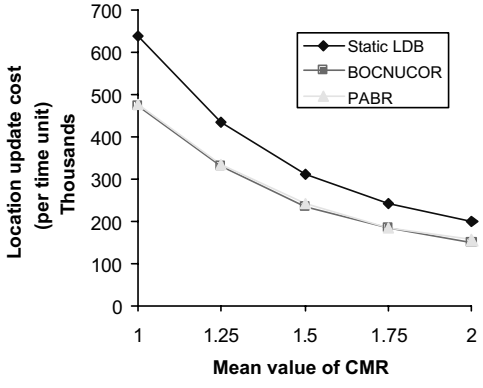

Fig. 5. Location update cost.

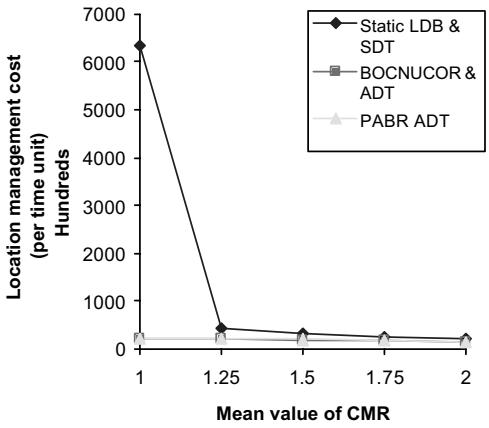

Fig. 6. Location update cost.

structures in organizing the location databases, both them could generate results close to the optimal one.

Fig. 6 shows the location update cost under the reorganization methods BOCNUCOR and PABR were used with ADT for generating location updates. Consistent with our expectation and the results shown in Fig. 5, the location update cost was smaller when the reorganization methods were applied. Furthermore, the reduction in update cost was great, compared with the case when SDT was used.

7. Conclusion

Maintaining the real-time locations of mobile clients is essential to many new mobile computing systems. In this paper, we study how to improve the system performance over a hierarchical location database structure when a distance-based location update policy is used for generating location updates. Although a hierarchical database structure is more scalable, compared with a two-tier location database structure, the cost for location update can be very high when a distance-based method is adopted. The cost for location update depends on the distance between the callers and the callees, and it can cause serious penalty to the system performance. In this paper, a cost-based method is proposed for deriving the optimal distance threshold. The objective is to minimize the total update cost and location management cost. Furthermore, under the existing hierarchical location database scheme, the tree structure of the location databases remains static. It cannot be adaptive to the changes on the mobility patterns of the mobile clients. To solve the problem, we present a reorganization strategy in the second part of the paper to restructure the hierarchical tree of location databases according to the mobility patterns of the clients. The objective is to minimize the location management cost. Simulation experiments have been performed to inves-

tigate the performance of the proposed method for deriving the optimal distance threshold and the reorganization strategy. Based on the results, we find out that the proposed methods can effectively reduce the location management cost.

References

Das, S.K., Sen, S.K., 1999. Adaptive location prediction strategies based on a hierarchical network model in a cellular mobile environment. The Computer Journal 42 (6), 473–486.

Gok, G., Ulusoy, Ö., 2000. Transmission of continuous query results in mobile computing systems. Information Sciences 125 (1–4), 37–63.

Levy, H., Naor, Z., 1999. Active tracking: locating mobile users in personal communication service networks. Wireless Network 5 (6), 467–477.

Mouly, M., Pautet, M.B., 1992. The GSM System for Mobile Communication, Cell and Systems, Telecom Publishing.

Pitoura, E., Fudos, I., 1998. An efficient hierarchical scheme for locating highly mobile users. In: Proceedings of the 6th ACM International Conference on Information and Knowledge Management (CIKM98), pp. 218–225.

Pitoura, E., Samaras, G., 2001. Locating objects in mobile computing. IEEE Transactions on Knowledge and Data Engineering 13 (4), 571–592.

Plassmann, D., 1994. Location management strategies for mobile cellular networks of 3rd generation. In: Proceedings of 44th IEEE Vehicular Technology Conference, pp. 649–653.

Rose, C., 1996. Minimizing the average cost of paging and registration: a timer-based method. Wireless Network 2 (2), 109–116.

Rose, C., Yates, R., 1996. Minimizing the average cost of paging under delay constraints. Wireless Network 2 (3), 109–116.

Wan, G., Lin, E., 1999. Cost reduction in location management using semi-realtime movement information. Wireless Network 5 (5), 245–256.

Xie, H., Tabbane, S., Goodman, D., 1993. Dynamic location area management and performance analysis. In: Proceeding of 43rd IEEE Vehicular Technology Conference, pp. 536–539.

Xiong, M., Ramamritham, K., Stankovic, J.A., Towsley, D., Sivasankaran, R., 2001. Scheduling transactions with temporal constraints: exploiting data semantic. IEEE Transactions in Knowledge and Data Engineering 14 (5), 1155–1166.

Li Guohui, he got his Ph.D. degree in 1999 in Huazhong University of Science and Technology (HUST) in China. Currently, he is an Associate Professor in the School of Computer Science and Technology in HUST. His main research interests include Active Database Systems, Real-Time Database Systems and Mobile Computing Systems.

Lam Kam-Yiu received the B.Sc. (Hons) degree in Computer Studies with distinction and Ph.D. degree from the City University of Hong Kong in 1990 and 1994, respectively. He is currently an Associate Professor in the Department of Computer Science, City University of Hong Kong. He has served as a program committee member and reviewer for conferences on real-time systems, mobile computing and databases, including the International Workshop on Real-time Databases, the International Workshops on Active, Real-time and Temporal Database Systems, the International Conference on Real-time Computing and Applications and the International Conference on Mobile Data Access. He is also a reviewer for international journals including the IEEE Transactions on Computers, IEEE Transactions on Data and Knowledge Engineering, IEEE Transactions on Parallel and Distributed Systems, Real-Time Systems, IEEE Multimedia, Journal of Real-Time Systems, The Computer Journal, the Information Processing Letters and the Journal of Systems and Software. His current research interests are real-time database systems, real-time active database systems, location–dependent continuous query processing, mobile computing and distributed multimedia systems. He was the Guest Editor of the Journal of Systems and Software on the special issue of Real-time Active Database Systems. He is a member of the IEEE.

Tei-Wei Kuo received B.S.E. degree in Computer Science and Information Engineering from National Taiwan University in Taipei, Taiwan, in 1986. He received the M.S. and Ph.D. degrees in computer sciences from the University of Texas at Austin in 1990 and 1994, respectively. He is currently a Professor in the Department of Computer Science and Information Engineering of the National Taiwan University, Taiwan, ROC. He was an Associate Professor in the Department of Computer Science and Information Engineering of the National Chung Cheng University, Taiwan, ROC, from August 1994 to July 2000. The research interest of him includes real-time databases, real-time process scheduling, real-time operating systems, and embedded systems. He is the Program Co-Chair of IEEE 7th Real-Time Technology and Applications Symposium, 2001, and an Associate Editor of the Journal of Real-Time Systems since 1998. He has consulted for government and industry on problems in various real-time systems design. He is a member of the IEEE computer society.

Shi-Wu Lo received the B.S.E. degree in Computer Science and Engineering from Yuan Ze University in Chungli, Taiwan, ROC, in 1998. He received the M.S. degree in Computer Science from the National Chung Cheng University, Taiwan, ROC, in 2000. He is currently a candidate for a doctor degree in the Department of Computer Science and Information Engineering of the National Taiwan University, Taiwan, ROC. His research interest is on real-time and embedded systems.