A Hierarchical Scheme for Locating Mobile Users

E. Pitoura and I. Fudos

97-02

Technical Report No. 97-02/1997

Department of Computer Science, University of Ioannina
GR 451 10 Ioannina, Greece, Tel./Fax +30-651-48131

A Hierarchical Scheme for Locating Mobile Users

Evaggelia Pitoura and Ioannis Fudos

Department of Computer Science

University of Ioannina

GR 45110 Ioannina, Greece

email: {pitoura,fudos}@cs.uoi.gr

Research Paper

Abstract

Locating moving objects is central to mobile computing. To reduce the cost of moves, instead of updating upon each move all location databases involved, a forwarding pointer is set to the new location. In this paper, we investigate the use of forwarding pointers in a hierarchical scheme of location databases. To avoid the building up of long chains of forwarding pointers, we propose various purging heuristics. We comparatively study the performance of the non forwarding and of the forwarding strategies along with a set of different purging heuristics for a range of call to mobility ratios and for users with different moving and calling behaviors.

Keywords: Mobile Computing, Distributed Database Systems, Simulation.

1 Introduction

In a Personal Communications Service (PCS) system, users place and receive calls through a wireless medium. PCS users are located in system-defined cells, which are bounded geographical areas [2, 9]. A cell is a uniquely identifiable unit. Inside a cell, a user can be tracked using some form of paging.

Databases are used to store information about the location of moving users. When user A places a call to user B, a number of database lookups are performed to identify the cell j where B resides. When user A enters a new cell, the location databases that contained information about A's old address must be updated.

For future PCS systems, with high user populations and numerous customer services, the signaling and database traffic for locating users is expected to increase dramatically [10]. To accommodate this increase in traffic, a distributed database architecture has been proposed in which location databases are organized in a tree [10, 1, 4]. Each location database contains information about the location of all users registered in levels below it. The use of a hierarchical scheme however, incurs considerable increases in the cost of move operations, since a number of location databases must be updated.

In this paper, we consider the problem of locating moving users in such hierarchical tree-structured location databases. In particular, to reduce the cost of moves, instead of updating all location databases involved, a forwarding pointer to the new location is set at the lower level database. However, if forwarding pointers are never deleted, then long chains are created, whose traversal results in an excessive increase in the cost of locating users during calls. Forwarding pointers have been also proposed for locating mobile users in [5, 3] but the work there is for non-hierarchical schemes. Forwarding pointers in a hierarchical database scheme of a different type is presented in [6]. However, the emphasis there is on adaptability rather than on strategies for purging forwarding pointers.

Purging forwarding pointers has many possible variations. We consider such variations and their relative performance. To study the performance of the forwarding scheme and of the various strategies for purging forwarding pointers, we have developed an event-driven simulator. We have performed a number of experiments for a range of call to mobility ratios, for users with different mobility and calling behavior, and for a real-word number of cells. The results clearly show that the forwarding scheme coupled with an appropriate purge heuristic on a per user basis can reduce the

cost of calls and moves by a factor of 2 for small call to mobility ratios.

The rest of this paper is organized as follows. In Section 2, we present the forwarding scheme and the various strategies for purging forwarding pointers. In Section 3, we describe our model for the call and mobility behavior of PCS users. In Section 4, we briefly describe our simulator and report performance results. Finally, in Section 5, we offer conclusions.

2 The Location Strategy

Databases are used to store information about the location of PCS users. These databases are interconnected by the links of the signaling network, e.g., a Common Channel Signaling network [8]. These databases form a tree that may be of arbitrary depth and width. The database at each leaf serves a single cell and contains entries for all users inside the cell it covers. A database at an internal node maintains information about all users registered in the set of cells in its subtree. Specifically, we assume that the entry for user A at a level i database contains a pointer to the level $i-1$ database in the subtree of which A resides. For example, in the tree hierarchy of Figure 1 for a user located at cell 25, there is an entry at the database at level 9 pointing to 25, at level 3 pointing to 9, and at 0 pointing to 3.

2.1 The basic location strategy

The problem of locating a mobile user can be described in terms of two basic operations, a call and a move operation. Consider PCS user A that *moves* from cell i to cell j. The entries along the path from i to the least common ancestor of i and j, denoted $LCA(i,j)$ must be updated. Specifically, a message propagates up the tree from j to $LCA(i,j)$ adding entries for A and then down the tree from $LCA(i,j)$ to i deleting the entries for A. For instance in Figure 1, if a user moves from cell 25 to cell 23, the entries at nodes 25 and 9 are deleted and entries for that user at 3, 8, and 23 are set.

Consider a call from PCS user A at cell i to user B located at cell j. The message from cell i

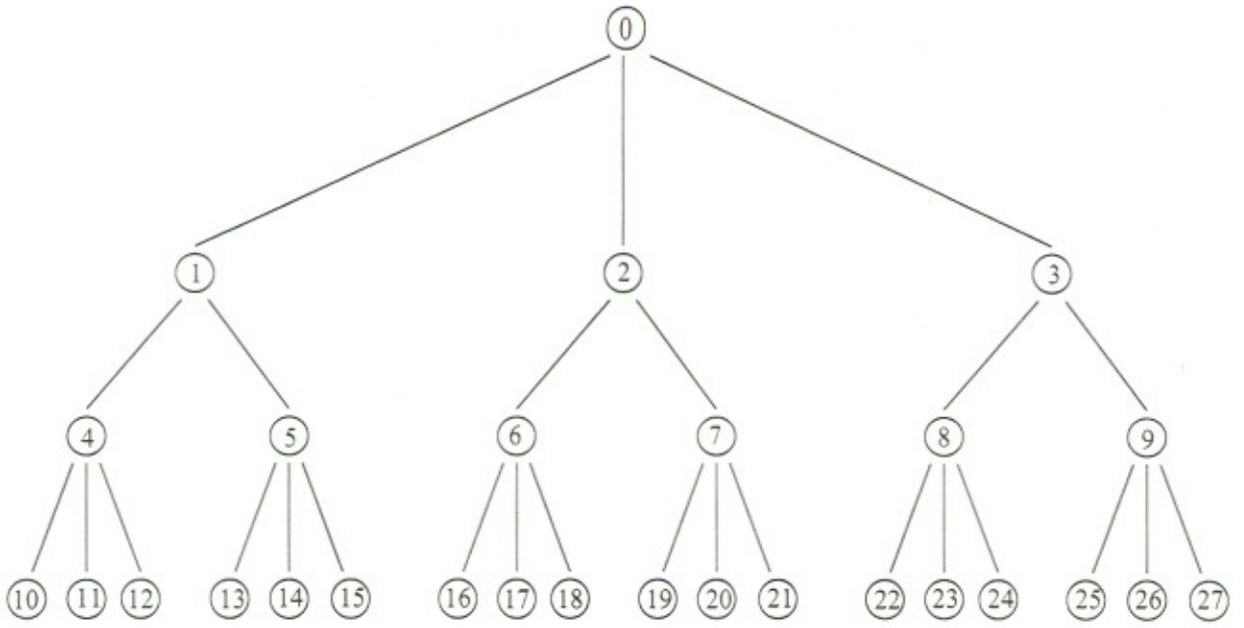


Figure 1: Hierarchical location scheme.

is propagated up the tree till an entry for user B is first found at $LCA(i,j)$. Then, the message propagates down the tree from $LCA(i,j)$ to the cell j where B resides. For instance, a call from a user at cell 26 to a user currently residing in 23, results in querying the databases at nodes 26, 9, and 3 and then at nodes 8 and 23.

We call the above operations *basic move* and *basic call* respectively.

2.2 The forwarding pointer location strategy

To cut down the cost of move operations, we propose using forwarding pointers. Upon each move from cell i to cell j, instead of updating all databases on the paths from j to $LCA(i,j)$ and from $LCA(i,j)$ to i, a forwarding pointer to j is set at the database of i. In the tree hierarchy of Figure 1, a move of PCS user A from cell 25 to 23, results simply in setting the entry for A at 25 to point to 23. No internal databases are updated.

Using forwarding pointers may increase the cost of calls. Calls originated from cell i to cell j proceed initially as in the basic case, going up to the $LCA(i,j)$ and then down to j. However, the

user may not be actually located at j, but the entree at j may be a forwarding pointer to another leaf node. Thus, before actually locating a user, a chain of forwarding pointers may have to be traversed. For instance, if a call is placed from a user at 22 to user A of the previous example, to locate A, databases at nodes 22, 8, 3, 9 and 25 are queried and then the forwarding pointer to 23 is followed.

2.3 Purging forwarding pointers

If forwarding pointers are never deleted, long chains of forwarding pointers are created resulting in calls with high latency. There are many variations in purging forwarding pointers depending on the form purge takes and on when it occurs.

In *simple purge*, we simply add a direct pointer from the first node of the chain to the current location of the user and delete all intermediate forwarding pointers. This results in a chain of length one. For example a chain $11 \rightarrow 18 \rightarrow 26 \rightarrow 14$ is replaced by chain $11 \rightarrow 14$. In *complete purge*, we also delete the entry in the first node of the chain. This causes the deletion of all entries at the internal databases from the first node to the LCA of the first node and the current location and the addition of entries on the path from the LCA to the current location. This results in a chain of zero length. For instance the chain $11 \rightarrow 18 \rightarrow 26 \rightarrow 14$ is deleted, as well as the entries at 4, and entries are set at 1, 5, and 14.

Purging of forwarding pointers, either simple or complete, can occur at calls or at moves. *Purging at calls* is initiated at each call when the first node of the chain is reached during the call. *Purging at moves* is initiated when a system-defined maximum on the length of the chain is reached. To purge pointers, the first node of the chain must be known. However, upon a move, this information is unavailable. Thus, to reach the first node, instead of keeping a single linked list of forwarding pointers, we maintain a doubly linked list, in which we also keep a backward pointer to the previous location of the user.

Finally, we note that before setting a forwarding pointer, detecting cycles is necessary to avoid infinite loops during calls. For example, consider chain $11 \rightarrow 18 \rightarrow 26 \rightarrow 14$ and a move made to

18, carelessly adding a pointer to 18 results in chain $11 \rightarrow 18 \rightarrow 26 \rightarrow 14 \rightarrow 18$ and future calls will hang. Cycles can be detected by checking upon each move of a user A from j to i, whether an entry for A already exists at i. If so, there is a path from j to i which can then be purged. Thus, the chain of the example becomes $11 \rightarrow 18$.

3 Modeling the Environment

3.1 The topology

We assume a hierarchy of location databases embedded in the hierarchy of telephone switches. To allow for maximum flexibility in the design of the location management scheme, we consider hierarchies with a variable number of levels h. The hierarchy of location databases is thus modeled by a complete tree of height h with an out degree $d(i)$ for level $i(i = 1 \ldots h)$. The leaves correspond to cells and thus each leaf corresponds to a unique physical address. In Figure 1, the location databases are organized in a complete tree of height 3 with $d(1) = 3$, $d(2) = 2$, and $d(3) = 3$.

3.2 Cost estimation

We assume that the physical address of a mobile user is the address of the cell inside which it currently resides. Tracking users inside a given cell is an issue orthogonal to identifying this cell and is beyond the scope of this paper. Let FF be the cost of following a forwarding pointer, i.e. sending a message to an arbitrary site knowing its physical address, and SF be the cost of setting a forwarding pointer. Similarly, let FL be the cost of following a link in the hierarchy of the location databases, i.e. sending a message to a parent or child location database site, and SF the cost of setting a link. Communicating with neighbors is less expensive than communicating with arbitrary cites. We have taken this into consideration when setting the relative values of SL, FL, FF, and SF in our simulation studies. In setting the values of SL, FL, FF, and SF, we also account for the mean delay in queues waiting to be served by a location database management system.

Let $current(x)$ be the cell where user x currently resides and k be the length of the chain of forwarding pointers. In the forwarding scheme, let $register(x)$ be the cell containing the first forwarding pointer.

We consider first a call placed by user x to user y.

- *Basic Schema:* In the basic scheme, the cost is

$$2\,lca(current(x), current(y))FL,$$

  where $lca(i,j)$ is the height of the $LCA(i,j)$ of leaf nodes i and j.

- *No purge:* In the forwarding schemes with no purge, the cost is

$$2\,lca(current(x), register(y))FL + kFF.$$

- *Simple Purge:* When simple purge of the chain of forwarding pointers takes place during calls, the overall cost is

$$2\,lca(current(x), register(y))FL + k(FF + SF).$$

- *Complete Purge:* When complete purge during calls takes place, in 1 to purging the forwarding chain, the location databases on the paths to the common ancestors of the location where the callee was first register and its current location are updated. Thus, the overall cost becomes:

$$2\,lca(current(x), register(y))FL + k(FF + SF) + 2\,lca(register(y), current(y))SL.$$

Let a move made from a user x to a new cell noted $new(x)$.

- *Basic Schema:* In the basic scheme the cost for a move is

$$2\,lca(current(x), new(x))(FL + SL).$$

- *Cycles:* In the forwarding schemes, the cost for a move if a cycle is detected in the forward chain is:

$$l\,SF + (l-1)FF,$$

where l is the length of the cycle.

- *No Purge (no cycles):* In the forwarding schemes with no purging the cost of a move is just the cost of setting a forwarding pointer, i.e.,

$$SF.$$

- *Simple Purge (no cycles):* When simple purge of the chain of forwarding pointers takes place during a move, the cost is

$$k(FF + SF).$$

- *Complete Purge (no cycles):* When complete purge during a move takes place the overall cost becomes

$$k(SF + FF) + 2\,lca(register(x), new(x))(SL + FL).$$

Finally, the probability of network contention in a hierarchical database arrangement is considered very small and in real world corresponds to temporary non-availability of the telephone network.

3.3 Calling and Mobility Model

We simulate calls to a specific mobile user and moves made by this user using an event-driven simulator. An event is either a move or a call event.

We assume that, for each user calls and moves occur independently. The interarrival times between two calls follow an exponential distribution, with parameter the mean interarrival time between two calls, t_c. The interarrival times between two moves follow another exponential distribution, with parameter the mean interarrival time between two moves, t_m. The ratio of the number of calls over the number of moves called *Call to Mobility Ratio* (CMR) is then

$$CMR = \frac{t_m}{t_c}.$$

The source of a call event is selected using one of the following distributions:

- **Arbitrary Calls**

  A call may be placed from any cell with equal probability $\frac{1}{n}$, where n is the number of different cells. We use a discrete uniform distribution to select one from the n cells.

- **Set of Frequent Callers**

  Each user receives most of its call for a specific set of locations. This corresponds to a real-life situation in which a user is frequently called by a set of other users or groups of users, e.g., friends, family, business associates or regular customers. We model a set of frequent callers with a discrete bimodal distribution, which distributes a 0.9 probability uniformly over a set of specific locations and a 0.1 probability uniformly over all other locations. So, a call has $\frac{0.9}{n_f}$ probability to be placed by a frequent calling location, and $\frac{0.1}{n-n_f}$ to be placed by another location, where n_f is the number of frequent calling locations.

The destination of a move event is selected via one of the following distributions:

- **Arbitrary Moves**

  A user may move to any location, except from its current, with the same probability. We use a uniform distribution as in the case of arbitrary calls, however the probability that the user remains in the same location after a move is 0, thus the probability that any other location is selected as the destination of the move is $\frac{1}{n-1}$.

- **Frequent Moves to Nearby Locations**

  Since users usually move to nearby locations, we model such a situation in which distant moves are unlikely to happen and short moves to neighbor locations are most likely to happen. We use

a discrete probability distribution in which the probability of a user moving to some location x decreases hyperbolically with the distance of x from the current location.

- **Movement by a Pattern**

  Users usually have a certain mobility pattern that is the same over short periods of time. For example, such a scenario corresponds to users that follow a daily routine, e.g., drive from their home to their office, visit a predetermined number of customers, return to their office, and then back to their home. In such cases, the user moves according to a certain pattern, e.g., among specific locations. This pattern can change but remains fixed for short periods of time (time locality). To model such scenarios, a sequence of locations is selected and the user moves circularly through that sequence.

To produce random numbers for the discrete probability distributions of the cases of set of callers and of frequent moves to nearby locations we have used the alias method (see e.g. [7]). The alias method models arbitrary discrete probability distributions with a fixed range.

4 Experiments and Results

We have developed an event-driven simulator to evaluate the performance of the location strategies. The simulator software has been developed in C++ and runs on a SUN 10 workstation.

We study the behavior of (1) the basic scheme, (2) the forwarding scheme with no purging of the forwarding pointers, (3) the forwarding scheme with simple purging at moves, (4) the forwarding scheme with complete purging at moves, (5) the forwarding scheme with simple purging at calls, and (6) the forwarding scheme with complete purging at moves. For purging at moves, the maximum length of the chain was set to 5, based on the analysis for the non hierarchical scheme in [5].

We run the experiments for a wide range of call to mobility ratios and for a total of 4000 move and call events. We present the results for three types of environments: (a) arbitrary calls and moves, (b) arbitrary calls and moves to nearby locations, and (c) arbitrary calls and pattern of moves.

We have also performed the same set of experiments for set of callers. As expected, the results are similar to the case of arbitrary calls, since what actually affects the performance of the scheme is the movement model. We plan to experiment with scenarios in which there is some correlation between the movement and calling behavior. For instance, when the set of callers has common sites with the moving pattern or the callers reside in nearby-locations. For example, that would be the case when the majority of calls to a PCS user originate from the town where the user usually resides.

We have experimented with hierarhies of different height and width. The results show that the relative performance of the schemes under consideration remains the same. The results presented are for a tree of height 10 and of a total number of about 17000 cells.

We estimate the benefits of forwarding by computing the ratio $\frac{Cost_{basic}}{Cost_{forward}}$ where for the cost of forward we use the cost of the simple purging at moves schemes, that proves to have a relative stable performance at all three types of environments. We use the results to draw conclusions about the appropriate forwarding scheme to be deployed on a per-user basis or more accurately for classes of users with specific call and mobility behaviors and CMR ratios.

4.1 Arbitrary Calls and Moves

When moves are arbitrary, the chain of forwarding pointers becomes very long, thus the performance of the no purging strategy degrades rapidly as CMR grows. Specifically, as shown in Figure 2(up) the no purging scheme for $0.1 < CMR < 10$ performs 2 to 5 times worse than the basic scheme. Only when moves are very infrequent ($CMR > 100$), the performance of no purging becomes comparable. The $\frac{Cost_{basic}}{Cost_{forward}}$ ratio for low call to mobility ratios ranges from 2 (at $CMR = 0.01$) to 1.7 (at $CMR = 1.0$). As expected for high CMRs, the benefits of forwarding become less apparent and the ratio drops to 1.2 at $CMR = 100$.

Figure 2(down) depicts the cost of the four purging strategies. For $CMR < 2$, the best strategy is the simple purge at calls. For $CMR > 2$, the simple or complete purge at moves strategies outperform the other strategies.

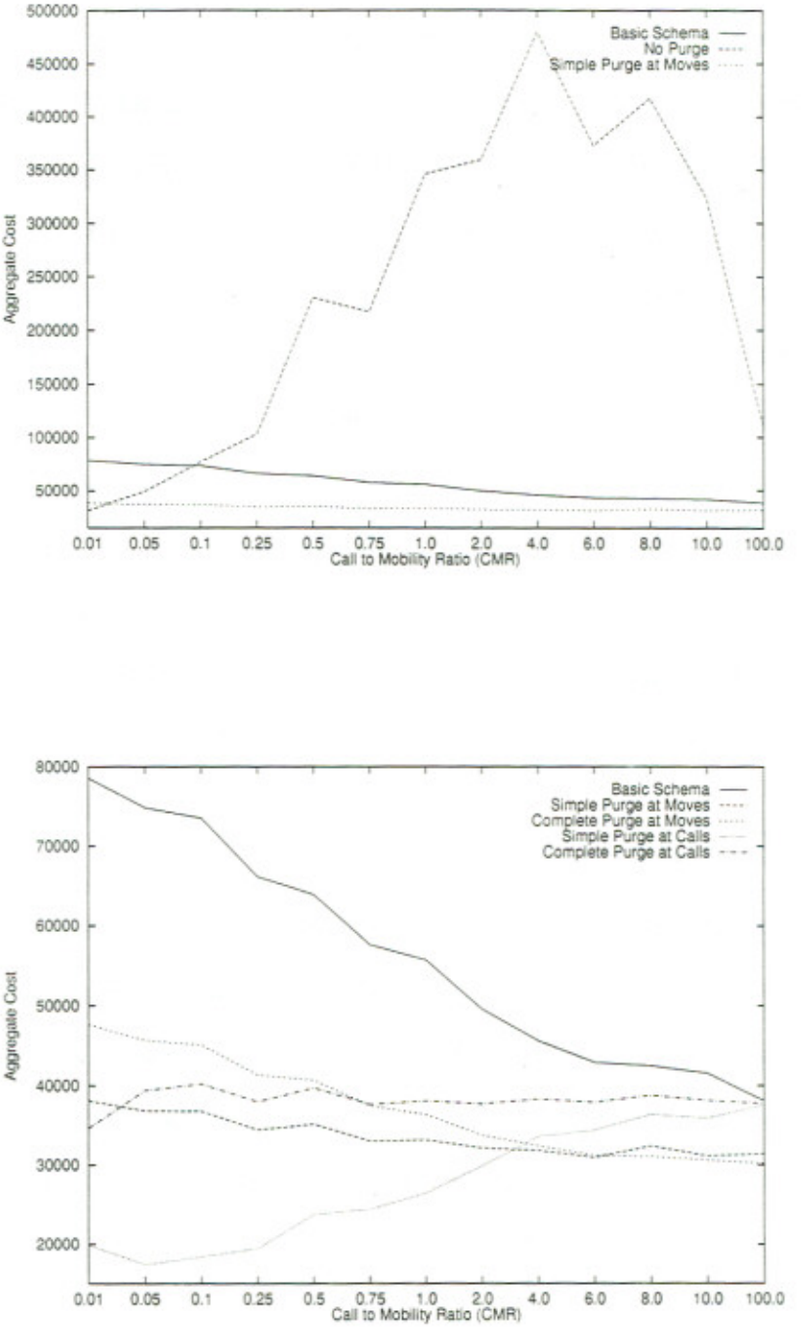


Figure 2: The behavior of the location schemes for arbitrary calls and moves.

4.2 Arbitrary Calls and Nearby Moves

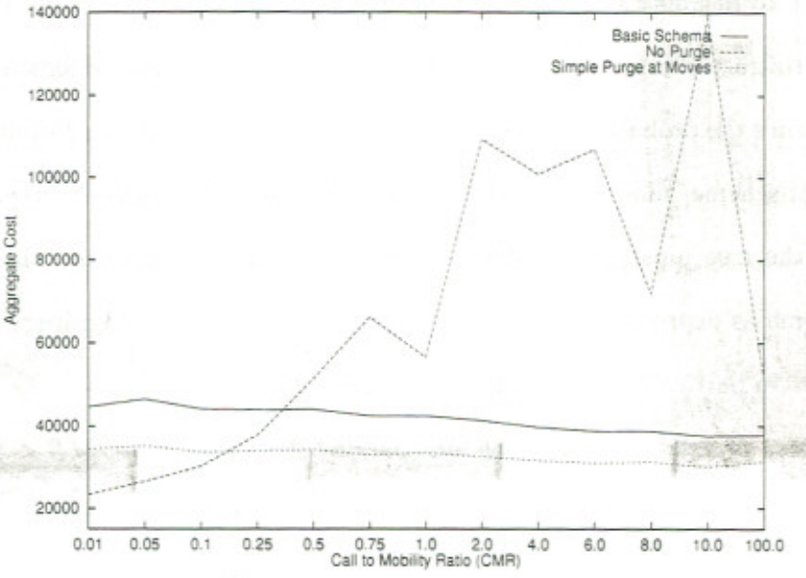
When moves are to neighbor cells, the cost of the no purging scheme is many times less than that in the case of arbitrary moves. This is because the length of the chain of forwarding pointers often remains short, since the probability of revisiting a site in the chain and thus forming a cycle increases. In a hierarchical scheme, moves to nearby location cost less than moves to remote locations since fewer levels in the tree must be updated. Thus, as expected in this case, the benefit of using a forwarding scheme as expressed by the ratio $\frac{Cost_{basic}}{Cost_{forward}}$ is of a lesser magnitude than in the case of arbitrary moves. In particular, it is around 1.3 (Figure 3(up)).

Figure 3(down) depicts the cost of the four purging strategies. For $CMR < 3$, the best strategy is the simple purge at calls. For $CMR > 3$, simple or complete purges at moves outperform the other strategies. Using the simple purge at calls strategy for low CMR values instead of the simple purge at moves strategy increases the $\frac{Cost_{basic}}{Cost_{forward}}$ ratio from around 1.3 to a range from 2.1 ($CMR = 0.01$) to 1.5 ($CMR = 1$).

4.3 Arbitrary Calls and Pattern of Moves

For the pattern of moves mobility model, we assume that the user moves among 10 arbitrarily chosen cells. Since cycles are detected, in this case, the length of the chain never exceeds 10. Thus, as expected the no purge forwarding scheme has comparable behavior with that of the other purge strategies (Figure 4(down)). Moreover, for $CMR < 1$ it performs better than the complete purging schemes and for $CMR < 0.75$ better than the simple purge at moves scheme. The $\frac{Cost_{basic}}{Cost_{forward}}$ ratio for low call to mobility ratios ranges from 2.3 (at $CMR = 0.01$) to 2 (at $CMR = 1.0$). As expected for high CMRs, the benefits of forwarding become less apparent and the ratio drops to 1.3 at $CMR = 100$ (Figure 4(up).

Figure 4(down) depicts the cost of the four purging strategies and that of the no purging scheme. For $CMR < 1$, the best strategy is the simple purge at calls. For $CMR > 1$, the simple purge at



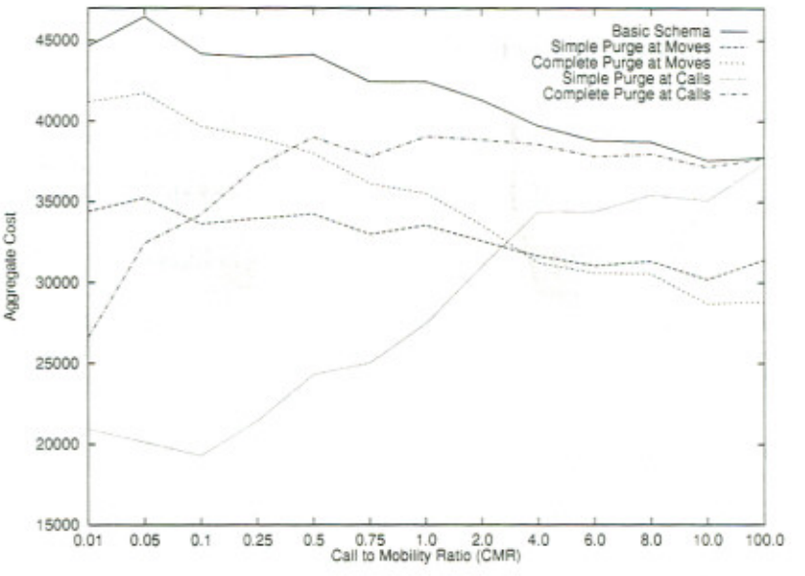
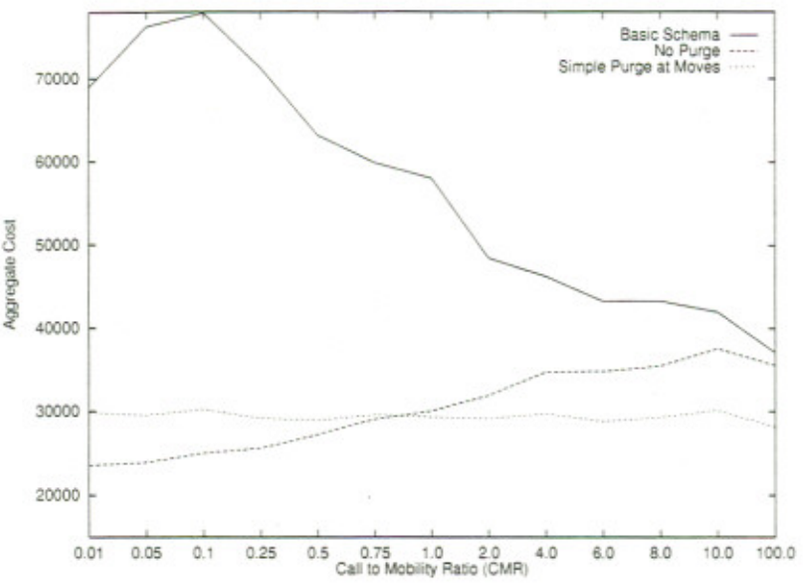


Figure 3: The behavior of location schemes for arbitrary calls and moves to nearby locations.



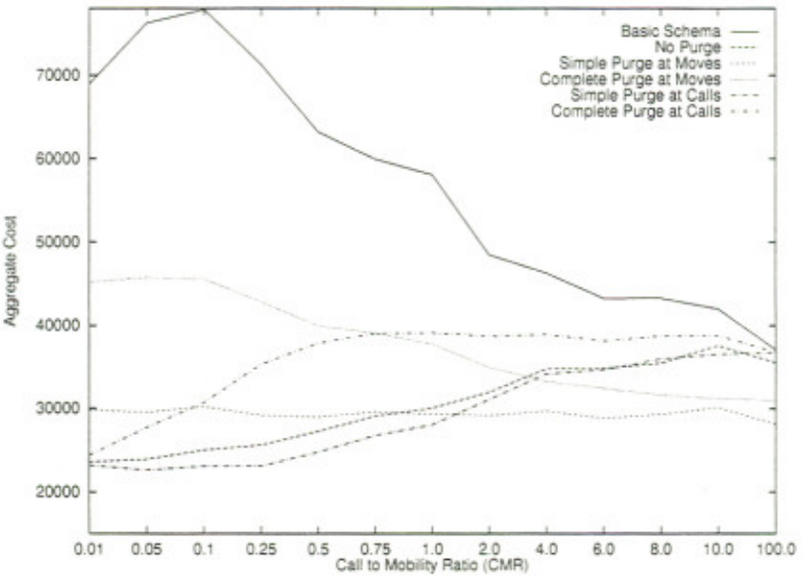


Figure 4: The behavior of location schemes for arbitrary calls and moves based on a pattern.