Tracking Mobile Users Using Hierarchical Location Databases

Evaggelia Pitoura and Ioannis Fudos
Department of Computer Science
University of Ioannina
GR 45110 Ioannina, Greece
email: {pitoura,fudos}@cs.uoi.gr

Abstract

To accommodate the increase in user population in future personal communication systems, hierarchical architectures of location databases have been proposed. In this paper, a scheme based on forwarding pointers is presented, that reduces the cost of network and database traffic generated by frequent location updates in such hierarchical architectures. To avoid the building up of long chains of forwarding pointers, various purging strategies are also introduced. We comparatively study the communication and database cost of the non forwarding and of the forwarding schemes along with a set of different purging strategies for a range of call to mobility ratios and for users with different moving and calling behaviors. An adaptive scheme is then proposed that takes advantage of these studies. Finally, we discuss consistency issues that arise when performing updates and lookups at the various levels of the distributed location database and for the case of concurrent arrivals of the call and move requests.

Keywords: Mobile Computing, Mobile Databases, Location Management, Pointer Forwarding.

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 [5, 14]. 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 [17]. To accommodate this increase in traffic, a distributed database architecture has been proposed in which location databases are organized in a tree [17, 1, 7]. 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.

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. Since whether the database or the communication cost is the dominating factor follows from various system-dependent parameters, we treat each of the costs separately in our analysis. 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 2-5 for small call to mobility ratios. We use the result of this analysis to outline an adaptive scheme that switches to the most appropriate strategy. Then, we propose an algorithm to synchronize the concurrent execution of call and move operations.

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 and in Section 5 we present an adaptive version of the scheme. In Section 6, we discuss concurrency issues. In Section 7, we compare our work with related research and in Section 8, we offer conclusions.

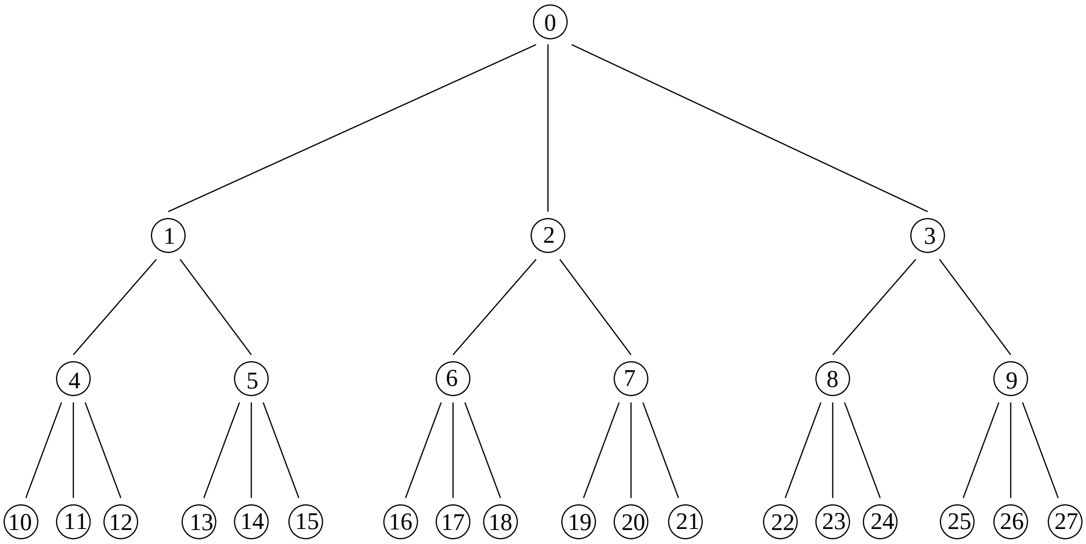


Figure 1: A hierarchical location database architecture.

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 [13]. 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 level 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 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 F be the cost of following a forwarding pointer, i.e. sending a message to an arbitrary site knowing its physical address, and L 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. Let U be the cost of a database update and Q the cost of a database query. Communicating with neighbors is less expensive than communicating with arbitrary sites. We have taken this into consideration when setting the relative values of F, and L in our simulation studies. In setting the values of U, and Q, 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

$$2\,lca(current(x), current(y))L+$$
$$(2\,lca(current(x), current(y)) + 1)Q,$$

  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

$$2\,lca(current(x), register(y))L + kF+$$
$$(2\,lca(current(x), register(y)) + k + 1)Q.$$

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

$$2\,lca(current(x), register(y))L + (k+1)F+$$
$$(2\,lca(current(x), register(y)) + k + 1)Q + kU.$$

- *Complete Purge:* When complete purge during calls takes place, in addition to purging the 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:

$$2\,lca(current(x), register(y))L + kF + 2\,lca(register(y), current(y))L+$$
$$(2\,lca(current(x), register(y)) + k + 1)Q + (2\,lca(register(y), current(y)) + k - 1)U.$$

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

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

$$2\,lca(current(x), new(x))L+$$

$$(2\,lca(current(x), new(x)) + 1)U.$$

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

$$(l-1)F + lU + lQ,$$

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.,

$$F + 2U.$$

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

$$(k+1)F + (k+2)U + (k+1)Q.$$

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

$$(k+1)F + 2\,lca(register(x), new(x))L+$$

$$(k+2)U + (k+1)Q + (2\,lca(register(x), new(x)) - 1)U.$$

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 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.

- **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. [12]). 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 [8].

We run the experiments for a wide range of call to mobility ratios and for a total of about 4000 move and call events per user. 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 hierarchies 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 for about 17000 cells.

Since the relative size of the communication and the database cost depends on many factors such as the load of the network and the size of the databases, we consider the two costs separately.

The experiments were performed multiple times (20) and statistical means were derived for all estimates.

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 best purging strategy at each case. 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 (see Figure 4(left)), thus the performance of the no purging strategy degrades rapidly as CMR grows. Only when moves are very infrequent ($CMR > 100$), the performance of no purging becomes comparable. The $\frac{CommunicationCost_{basic}}{CommunicationCost_{forward}}$ ratio for low call to mobility ratios ranges from 8.67 (at $CMR = 0.01$) to 2.10 (at $CMR = 0.5$) and drops to 1.05 (at $CMR = 6$). The $\frac{DatabaseCost_{basic}}{DatabaseCost_{forward}}$ ratio for low call to mobility ratios ranges from 5.5 (at $CMR = 0.01$) to 2.50 (at $CMR = 0.5$) and drops to 1.12 (at $CMR = 6$). As expected for high CMRs, the basic scheme outperforms all forwarding strategies except from the full purge at calls that has similar behavior with the basic scheme.

Figure 2(right) and Figure 3(right) depict the communication and database cost respectively of the basic scheme and the four purging strategies. For $CMR < 10$ the best strategy in terms of database cost is the simple purge at calls. For $CMR < 8$ the best strategy in terms of communication cost is the simple purge at calls.

The mean chain length (Figure 4(right)) is almost independent of the CMRs except for the case of purge at calls and for really small values of the CMR. In this case, the chain grows very long since the number of moves between two consequent calls is large.

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 (Figure 7). The same holds for the other purging strategies. 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, the $\frac{CommunicationCost_{basic}}{CommunicationCost_{forward}}$ ratio for low call to mobility ratios ranges from 5.5 (at $CMR = 0.01$) to 1.72 (at $CMR = 0.5$) and drops to 1.08 (at $CMR = 2$). The $\frac{DatabaseCost_{basic}}{DatabaseCost_{forward}}$ ratio for low call to mobility ratios ranges from 3.51 (at $CMR = 0.01$) to 1.91 (at $CMR = 0.5$) and drops to 1.16 (at $CMR = 2$).

Figure 5(right) and Figure 6(right) depict the communication and database cost respectively of the basic and the four purging strategies. For $CMR < 6$ the best strategy in terms of database cost is the simple purge at calls or moves. For $CMR < 4$ the best strategy in terms of communication cost is the simple purge at calls or moves.

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 9 (Figure 10(left)). Thus, as expected the no purge forwarding scheme has comparable behavior with that of the other purge strategies (Figure 8(left) and Figure 9(left)). The $\frac{CommunicationCost_{basic}}{CommunicationCost_{forward}}$ ratio for low call to mobility ratios ranges from 14 (at $CMR = 0.01$) to 2.17 (at $CMR = 0.5$) and drops to 1.13 (at $CMR = 4$). The $\frac{DatabaseCost_{basic}}{DatabaseCost_{forward}}$ ratio for low call to mobility ratios ranges from 5.69 (at $CMR = 0.01$) to 2.61 (at $CMR = 0.5$) and drops to 1.26 (at $CMR = 4$).

Figure 8(right) and Figure 9(right) depict the communication and database cost respectively of the basic and the four purging strategies. For $CMR < 10$ the best strategy is the

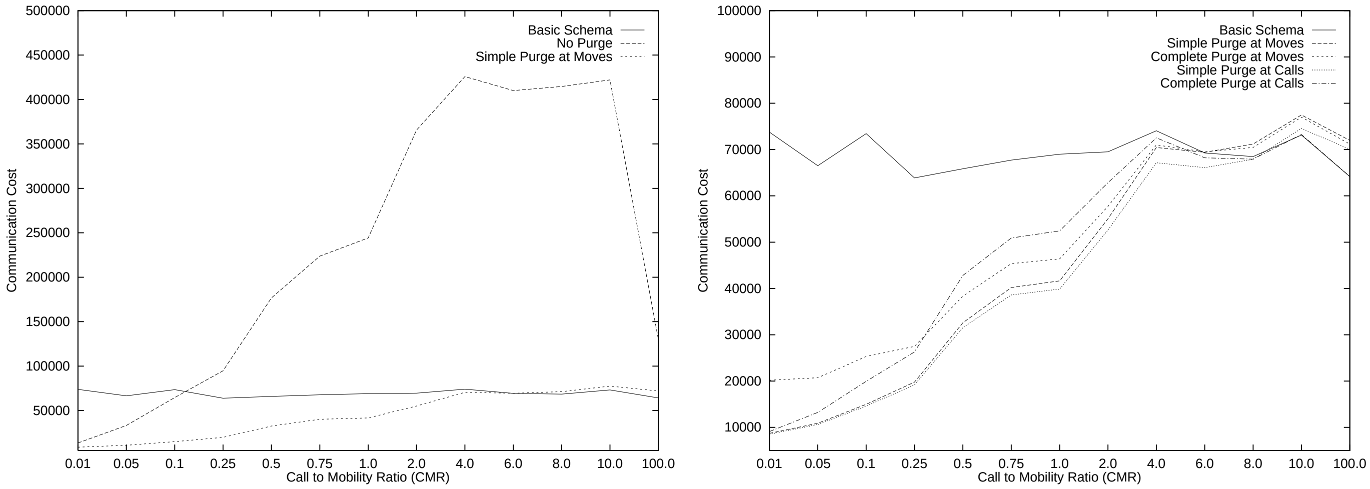


Figure 2: Communication cost for arbitrary calls and moves.

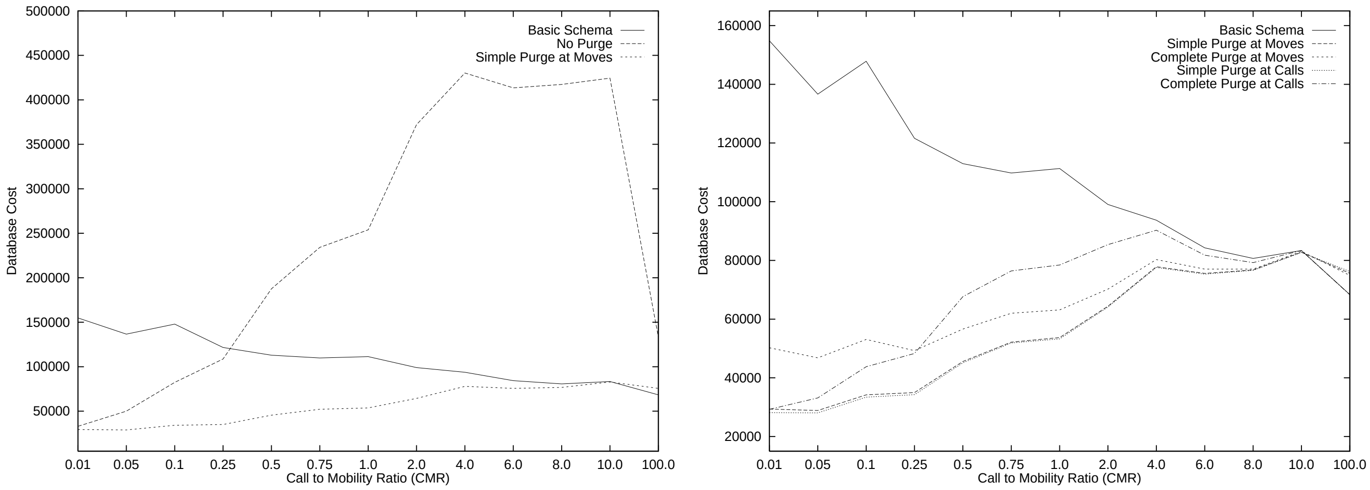


Figure 3: Database cost for arbitrary calls and moves.

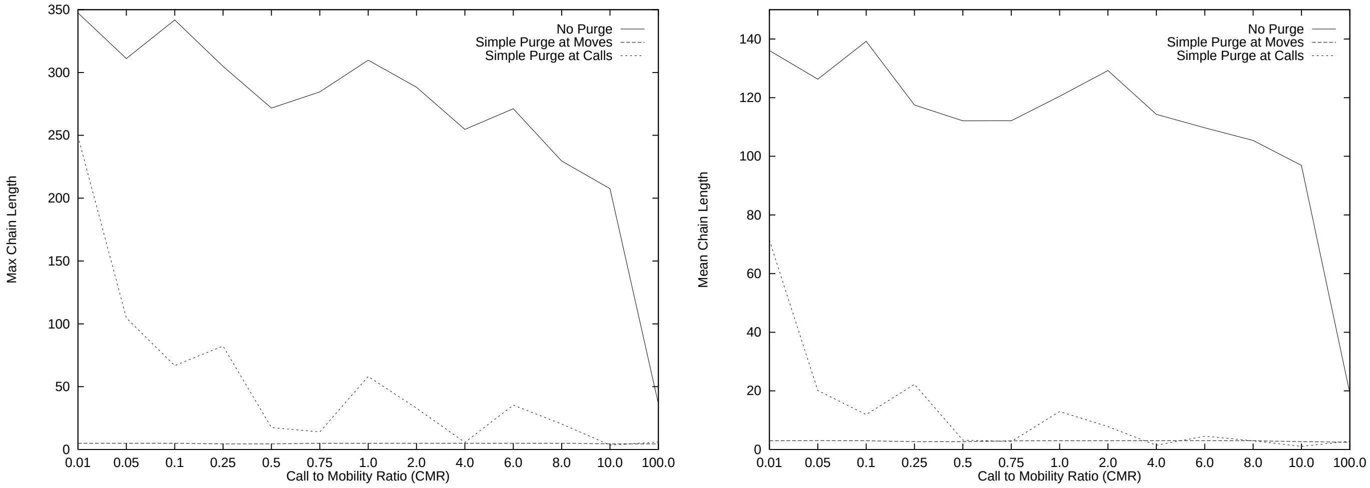


Figure 4: (left) Maximum length and (right) mean length of the forwarding chain for arbitrary calls and moves.

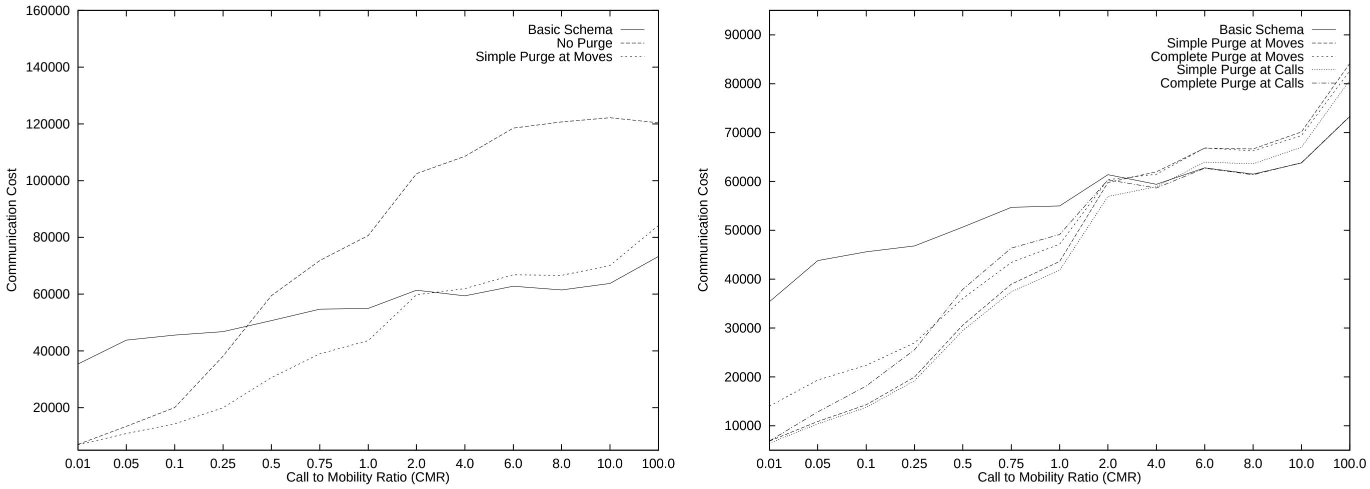


Figure 5: Communication cost for arbitrary calls and moves to nearby locations.

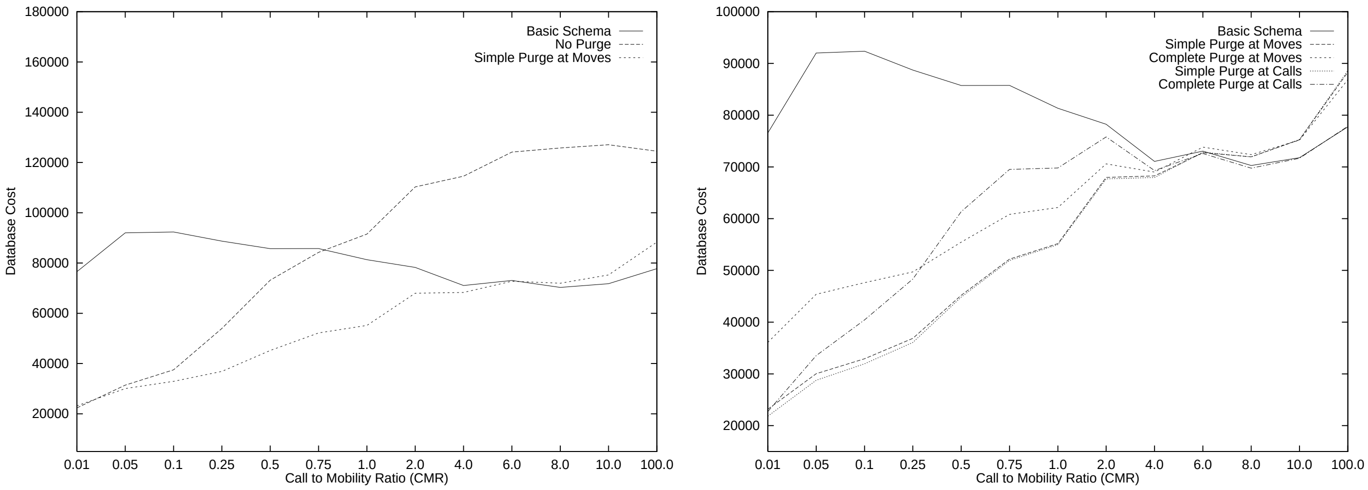


Figure 6: Database cost for arbitrary calls and moves to nearby locations.

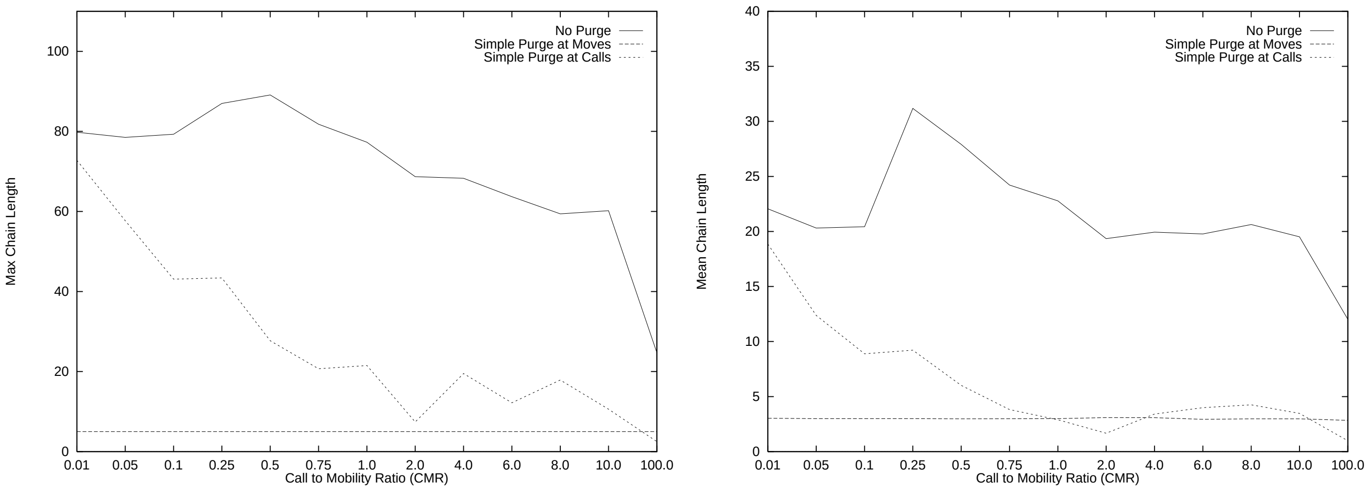


Figure 7: (left) Maximum length and (right) mean length of the forwarding chain for arbitrary calls and moves to nearby locations.

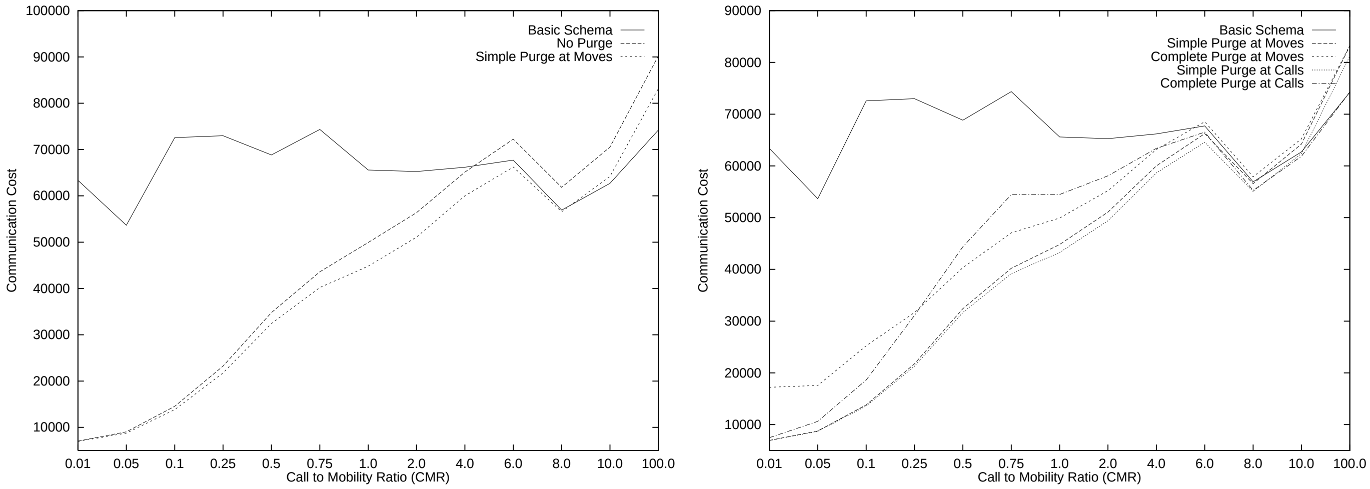


Figure 8: Communication cost for arbitrary calls and moves following a pattern.

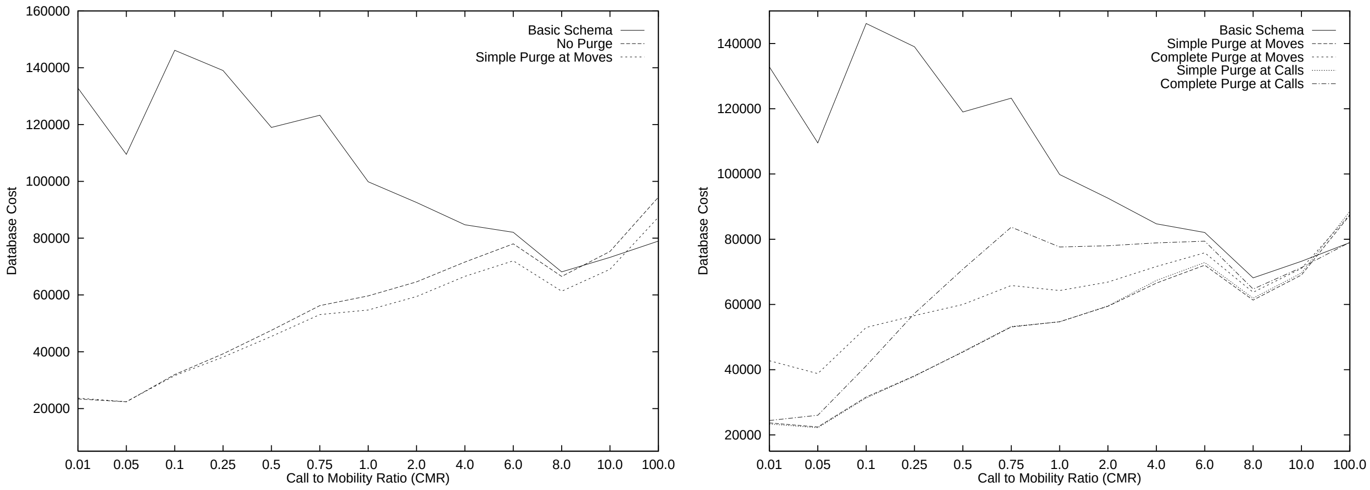
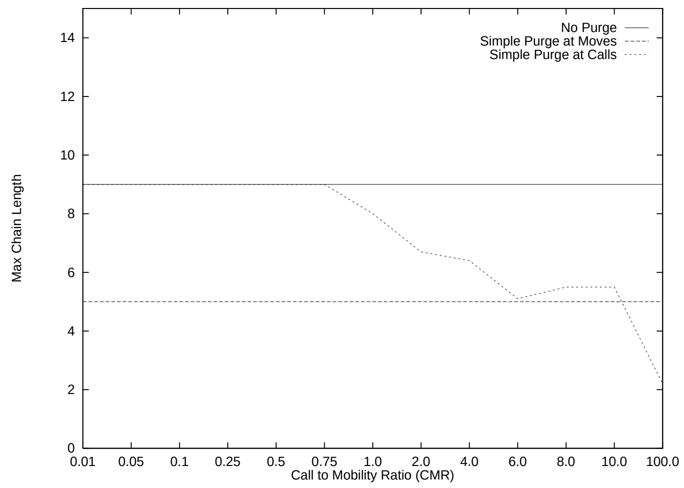


Figure 9: Database cost for arbitrary calls and moves following a pattern.



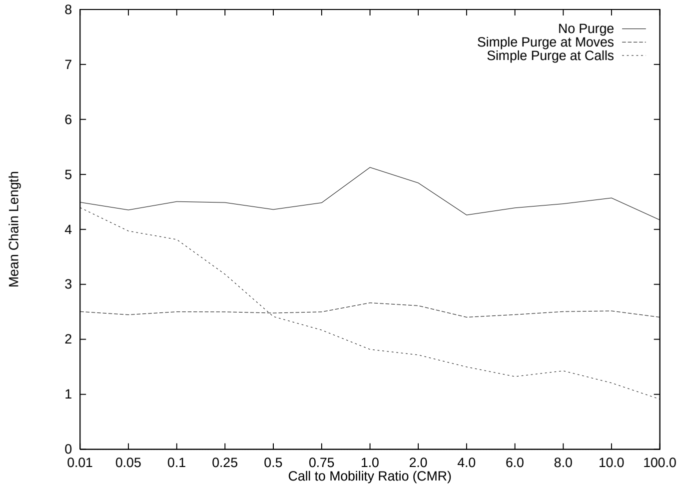


Figure 10: (left) Maximum length and (right) mean length of the forwarding chain for arbitrary calls and moves following a pattern.

simple purge at calls.

5 An Adaptive Strategy

We use the results of the simulation to design an effective adaptive version of the forwarding scheme. The adaptive version is deployed on a per user basis.

The call to mobility ratio for a specific user can be either given as part of the user's profile or it can be estimated using an appropriate on-line estimation algorithm, such as the running average or the reset k-algorithm described in [9]. The moving behavior and the caller set of each user are captured by monitoring its moving and calling history. Then, the appropriate forwarding scheme is chosen among the basic, no purge, simple purge at calls, simple purge at moves, complete purge at calls or complete purge at moves strategies. The scheme is recorded at the database of the current location of the user. The scheme may change when either the CMR or the mobility behavior of the user changes.

In selecting the most appropriate forwarding scheme, a decision table is utilized. The decision table can be precomputed for a specific location database hierarchy, or for parts of the hierarchy that are characterized by small in between traffic. The table is independent of the database load and the network traffic, since the results are drawn based on the comparative behavior of the schemes rather than on their absolute costs.

The analysis of Sections 3 and 4 are applicable to any type of hierarchical database architecture. For demonstration purposes, we use the analysis of Section 3 and the simulation results of Section 4, to construct the decision table for a location architecture with 10 levels and around 17000 leaf nodes. When one of the communication and database costs clearly dominates the other, then the table is constructed based on the dominating cost. When none of the two costs clearly dominates the other, strategies that perform well in terms of both costs are preferred. For example, when no cost clearly dominates, the decision table is illustrated in Figure 11. When the database cost dominates, the table of Figure 12 is used. Finally, when the communication cost dominates the database cost, the decision table of Figure 13 is used.

When a call succeeds in finding the current location of a user, the location database at this site is advised to determine which purging strategy to apply. If a scheme that does not require purging at calls has been selected no action takes place. Otherwise, a complete or simple purge is performed. Likewise, when a user moves, the database at its current location is first examined, and the type of purging is decided.

6 Concurrency

In the treatment of the problem, we have not so far considered concurrency issues. Concurrency problems arise in any hierarchical location database. A call placed by user A to user B results in a query operation being issued at each database on the path to their least common ancestor. Similarly, a move made from location i to location j causes an update operation to be executed at each database on the path to their least common ancestor. The discussion was based on the assumption that moves and calls arrive sequentially and are handled atomically and isolated one at a time. Thus, it was assumed that there is no interleaving between the queries and the updates of the various call and move operations. This is a reasonable assumption only if all network and database operations are performed in negligible time.

To tackle the problem, we consider three different approaches. In any case, a forwarding pointer to the new location must be set at the old address. Otherwise calls that have been issued prior to the movement and thus arrive at the old address would be lost.

In the *lazy and lossy* approach, no special treatment is given to concurrent operations. Calls that read obsolete data fail to track the user, and the lookup procedure is reissued anew. This method though simple, does not provide any upper bound on the number of tries a pending call, i.e., a call that had been made before the user moved, has to make before

ENVIRONMENT \ CMR RANGES	CMR<2	2<CMR<4	4<CMR<6	6<CMR<8	8<CMR<10	10<CMR<100
ARBITRARY CALLS AND MOVES	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	COMPLETE AT CALLS	BASIC
ARBITRARY CALLS AND MOVES TO NEARBY CELLS	SIMPLE AT CALLS	SIMPLE AT CALLS	COMPLETE AT CALLS	COMPLETE AT CALLS	COMPLETE AT CALLS	BASIC
ARBITRARY CALLS AND PATTERN OF MOVES	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	COMPLETE AT CALLS
SET OF CALLERS AND ARBITRARY MOVES	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	COMPLETE AT CALLS	BASIC
SET OF CALLERS AND MOVES TO NEARBY CELLS	SIMPLE AT CALLS	SIMPLE AT CALLS	COMPLETE AT CALLS	COMPLETE AT CALLS	COMPLETE AT CALLS	BASIC
SET OF CALLERS AND PATTERN OF MOVES	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	COMPLETE AT CALLS

Figure 11: Decision table for the example architecture.

ENVIRONMENT \ CMR RANGES	CMR<2	2<CMR<4	4<CMR<6	6<CMR<8	8<CMR<10	10<CMR<100
ARBITRARY CALLS AND MOVES	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	BASIC
ARBITRARY CALLS AND MOVES TO NEARBY CELLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	COMPLETE AT CALLS	COMPLETE AT CALLS	BASIC
ARBITRARY CALLS AND PATTERN OF MOVES	SIMPLE AT CALLS	SIMPLE AT MOVES	SIMPLE AT MOVES	SIMPLE AT MOVES	SIMPLE AT MOVES	COMPLETE AT CALLS
SET OF CALLERS AND ARBITRARY MOVES	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	BASIC
SET OF CALLERS AND MOVES TO NEARBY CELLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	COMPLETE AT CALLS	COMPLETE AT CALLS	BASIC
SET OF CALLERS AND PATTERN OF MOVES	SIMPLE AT CALLS	SIMPLE AT MOVES	SIMPLE AT MOVES	SIMPLE AT MOVES	SIMPLE AT MOVES	COMPLETE AT CALLS

Figure 12: Decision table based on the database cost for the example architecture.

ENVIRONMENT \ CMR RANGES	CMR<2	2<CMR<4	4<CMR<6	6<CMR<8	8<CMR<10	10<CMR<100
ARBITRARY CALLS AND MOVES	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	BASIC	BASIC
ARBITRARY CALLS AND MOVES TO NEARBY CELLS	SIMPLE AT CALLS	SIMPLE AT CALLS	COMPLETE AT CALLS	BASIC	BASIC	BASIC
ARBITRARY CALLS AND PATTERN OF MOVES	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	COMPLETE AT CALLS
SET OF CALLERS AND ARBITRARY MOVES	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	BASIC	BASIC
SET OF CALLERS AND MOVES TO NEARBY CELLS	SIMPLE AT CALLS	SIMPLE AT CALLS	COMPLETE AT CALLS	BASIC	BASIC	BASIC
SET OF CALLERS AND PATTERN OF MOVES	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	SIMPLE AT CALLS	COMPLETE AT CALLS

Figure 13: Decision table based on the communication cost for the example architecture.

locating a moving user. In the *transactional* approach, traditional database concurrency control techniques are used to enforce that each call and move operation is executed as a transaction, i.e., an isolated unit. This approach is highly impractical since for instance acquiring locks at all distributed databases involved in a call or move operation causes prohibitive delays.

The third and most practical approach is based on imposing *a specific order* on the way updates are performed. In particular, upon a move operation from i to j, first entries at the path from j to $lca(i,j)$ are added in a bottom-up fashion and then the entries at the path from the $lca(i,j)$ to i are deleted in a top-down fashion. In a similar manner, forwarding pointers are purged starting from the first entry in the chain and moving towards the last entry. Thus, moves have two phases: an add phase and a delete phase. Calls also have two phases: an upward and a downward phase. During the downward phase, a call moves up the tree to find an entry for the callee. Once the first entry for the location of the callee is found, the call moves downwards to this location.

In addition, we associate with each entry for x in each location database at level k a counter, called pending calls counter (PCC_k). The PCC counts the number of calls that have queried this entry while moving downwards to locate x. Note, that $PCC_k > PCC_{k-1}$ indicates that there is at least one call operation traveling from level k to level $k-1$. During the delete phase of a move operation, an entry at a level $k-1$ database is deleted only if $PCC_{k-1} \geq PCC_k$. Otherwise, the delete operation is delayed till the calls searching for x arrive at the level $k-1$ database.

A call c is concurrent with a move m, if a query operation of c is executed before some operation of m and after some other operation of m. In other words, if both c and m are in progress at the same time. We will prove that any call c concurrent with a move from i to j will succeed in locating the user.

Specifically, we will show that (i) during the upward phase, call c is directed either to the old location, i, or the new location j, and (ii) during the downward phase, c moves correctly to the location found in the first phase. Thus, c arrives at j either directly or indirectly by following the pointer found at i.

Proof of claim(i). Let a call c placed by user at cell o be at its upward phase and m be the node at level k where it encounters the first entry for x. This entry will be either a newly added entry if node m is on the path from j to $lca(i,j)$ and it has already been updated, or an entry directing the call to the old location. In any case, $m \leq max\{LCA(o,j), LCA(o,i)\}$.

Proof of claim(ii). Let a call c be at level k during its downwards phase after having found an entry for x. If the entry for x directs the call to the new location j, then the call will succeed in finding an entry at level $k-1$ since in the add phase the entries are included in a bottom-up fashion. If the entry directs the call to the old location i, the call will succeed finding an entry at level $k-1$ since an entry is deleted only when all pending calls from previous levels have been serviced as indicated by the PCC. Similar claims hold for following the forward chain, if any.

Thus, all calls will eventually succeed in reaching either i or j. In effect, the algorithm in the first case makes a call appears as if it has occurred before the move, that is, it pushes the call backwards, or makes it appear as if it has occurred after the move, that is it pushes the call forwards.

7 Related Work

Location management has attracted much current research in mobile computing. Various approaches have been proposed for reducing the cost of move and call operations for both hierarchical and non hierarchical location database architectures.

One proposal to reduce the cost of calls is replicating (see e.g., [16, 15] for hierarchical and [10] for non hierarchical architectures) or caching (see e.g., [9] for non hierarchical and [7] for hierarchical architectures) the location of mobile users at the sites of their frequent callers. To reduce the cost of network traffic generated by frequent updates, a scheme based on partitioning is presented in [3, 4]. Partitioning is performed by grouping the cells among which a mobile unit moves frequently and separating the cells between which it relocates infrequently. Only moves across partitions are reported. Finally, the assignment of location databases to the nodes of a signaling network is discussed in [1], where database placement is formulated as an optimization problem. Those schemes are orthogonal to the address forwarding strategy and can be used in conduction with it to further reduce database and communication costs.

Address forwarding for a non hierarchical architecture is discussed in [8] along with an analysis of its benefits for different CMRs. Forwarding pointers have been also used in [6] again for non hierarchical location databases. The focus there is on IP-based protocols. In [2], forwarding is explored in a different framework for a hierarchical, though not tree-structured, location database architecture, called regional matching directory. The treatment is theoretical and based on a worst case order analysis.

In [11], forwarding pointers are used in hierarchical location databases. However, the architectural assumptions are different in that the actual location is saved at each internal database instead of a pointer to the corresponding lower level database. The forwarding pointer is set at different levels in the hierarchy and not necessarily at the lower level database as in our scheme. The emphasis there is on choosing an appropriate level for setting the forwarding pointers and on updating obsolete entries in the hierarchy after a successful call.

8 Summary

Tracking mobile users is central to mobile computing. In this paper, we have studied the use of forwarding pointers in a hierarchical database arrangement embedded in a telephone network. We have comparatively studied the performance of non forwarding and of forwarding schemes for a variety of call to mobility ratios and mobility and calling behaviors. 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 2-5 when the call to mobility ratio is small. Finally, we have described a synchronization method to control the concurrent execution of call and move operations.

References

[1] V. Anantharam, M. L. Honig, U. Madhow, and V. K. Kei. Optimization of a Database Hierarchy for Mobility Tracking in a Personal Communications Network. *Performance Evaluation*, 20:287–300, 1994.

[2] B. Awerbuch and D. Peleg. Concurrent Online Tracking of Mobile Users. In *Proceedings of SIGCOMM 91*, pages 221–233, November 1991.

[3] B. R. Badrinath, T. Imielinski, and A. Virmani. Locating Strategies for Personal Communications Networks. In *Proceedings of the 1992 International Conference on Networks for Personal Communications*, 1992.

[4] G. Cho and L. F. Marshall. An Efficient Location and Routing Schema for Mobile Computing Environments. *IEEE Journal on Selected Areas in Communications*, 13(5), June 1995.

[5] T. Imielinksi and B. R. Badrinath. Wireless Mobile Computing: Challenges in Data Management. *Communications of the ACM*, 37(10), October 1994.

[6] J. Ioannidis, D. Duchamp, and G. Q. Maguire Jr. IP-based Protocols for Mobile Internetworking. In *Proceedings of the ACM SIGCOMM Symposium on Communications, Architectures and Protocols*, pages 235–245, September 1991.

[7] R. Jain. Reducing Traffic Impacts of PCS Using Hierarchical User Location Databases. In *Proceedings of the IEEE International Conference on Communications*, 1996.

[8] R. Jain and Y-B. Ling. A Auxiliary User Location Strategy Employing Forwarding Pointers to Reduce Network Impacts of PCS. *Wireless Networks*, 1:197–210, 1995.

[9] R. Jain, Y-B. Ling, C. Lo, and S. Mohan. A Caching Strategy to Reduce Network Impacts of PCS. *IEEE Journal on Selected Areas in Communications*, 12(8):1434–44, October 1994.

[10] J. Jannink, D. Lam, N. Shivakumar, J. Widom, and D. C. Cox. Efficient and Flexible Location Management Techniques for Wireless Communication Systems. In *Proceedings of the 2nd ACM International Conference on Mobile Computing and Networking (Mobicom'96)*, 1996.

[11] P. Krishna, N. H. Vaidya, and D. K. Pradhan. Static and Dynamic Location Management in Mobile Wireless Networks. *Journal of Computer Communications (special issue on Mobile Computing)*, 19(4), March 1996.

[12] R. A. Kronmal and A. V. Peterson Jr. On the Alias Method for Generating Random Variables from a Discrete Distribution. *American Statistician*, 33:214–218, 1979.

[13] S. Mohan and R. Jain. Two User Location Strategies for Personal Communication Services. *IEEE Personal Communications*, 1(1):42–50, 1st Quarter 1994.

[14] E. Pitoura and B. Bhargava. Building Information Systems for Mobile Environments. In *Proceedings of the Third International Conference on Information and Knowledge Management*, pages 371–378, Washington, D.C., November 1994.

[15] S. Rajagopalan and B. R. Badrinath. An Adaptive Location Management Strategy for Mobile IP. In *Proceedings of the 1st ACM International Conference on Mobile Computing and Networking (Mobicom'95)*, October 1995.

[16] N. Shivakumar and J. Widom. User Profile Replication for Faster Location Lookup in Mobile Environments. In *Proceedings of the 1st ACM International Conference on Mobile Computing and Networking (Mobicom'95)*, 161-169, October 1995.

[17] J. Z. Wang. A Fully Distributed Location Registration Strategy for Universal Personal Communication Systems. *IEEE Journal on Selected Areas in Communications*, 11(6):850–860, August 1993.