

# Reducing traffic impacts of PCS using hierarchical user location databases

Ravi Jain\*

Applied Research, Bellcore, Morristown, NJ 07960

**Abstract.** To accommodate the large PCS user populations expected in the future, mobility management architectures in which location databases are organized in a tree have been proposed, e.g. for third-generation mobile systems such as the Universal Mobile Telecommunication System (UMTS). We propose the use of caching as an auxiliary strategy for reducing the network impacts of locating mobile users in a tree-structured architecture. We discuss several issues which arise for implementing caching strategies in such an architecture. We quantify the costs and benefits of caching for one particular variation of a caching strategy in order to illustrate the trade-offs involved, and present possible alternative strategies also. We show that under certain assumptions, with an eager caching strategy where the signaling architecture is a tree, for users whose regional call-to-mobility ratio (RCMR) exceeds 5, caching can result in up to 30% reductions in total network cost as well as call setup time.

## 1 Introduction

For future PCS systems, with high user populations and numerous services available to users, it is expected that there will be a considerable increase in the amount of data to be managed, as well as signaling and database traffic [9, 11]. Thus a distributed database architecture has been proposed in which the databases are organized in a tree [10, 13, 1, 4], and is under study for the European third-generation mobile system called the Universal Mobile Telecommunication System (UMTS) [4].

A tree-structured location database architecture helps to reduce network signalling traffic when most calls received by users and most registration area crossings are geographically localized. However, the total number of databases queried, and hence call setup time, can potentially increase. We will illustrate this after defining the network model in sec. 2 and the basic user location strategies used in tree-structured

architectures in sec. 3. We have proposed that the judicious use of caching in tree-structured architectures can help alleviate this problem [5].

In tree location strategies [13, 1], as in most other PCS user location strategies [8], the same user location procedure has to be invoked for every call to a mobile user. The key observation we make is that, in many cases, it should be possible to re-use the information about the user's location from the previous call to that user. Clearly, this information will be useful for those users who receive calls frequently relative to the rate at which they cross registration areas, i.e., users with a high *call-to-mobility ratio* (CMR). This idea is essentially a form of *caching*, and in a previous study [7] we have investigated its benefits for a two-level location strategy, similar to that specified in the North American cellular IS-41 standard [3] and GSM [12]. We call the use of caching an *auxiliary* strategy; for further discussion, see [7].

Caching in tree architectures has many possible variations. We discuss its use in sec. 4 and describe one particular variation in detail. In sec. 5 we present a simple analysis of the costs and benefits of this strategy, and in sec. 6 we discuss other possible variations of caching.

## 2 System model

The basic system model we use is similar to one presented earlier [7]. Essentially, PCS cells are aggregated into contiguous geographical regions called Registration Areas (RA). For our purposes, determining the location of a user consists of finding the RA which the user is currently registered in. Databases are used to store information about the location of PCS users, and the databases are interconnected by the links of the signalling network, e.g. a Common Channel Signalling (CCS) network.

In the tree-structured architecture, the user location databases and CCS network form a tree. (See Fig. 1.) Such a system (or variations of it) has been proposed in [13, 1, 4].

The database at each leaf of the tree serves a single RA. It maintains the identities of the users currently registered in the RA it serves, as well as any

---

\*Address for correspondence: Bellcore, 445 South St, Morristown, NJ 07960. Phone: (201)-829-4756. Fax: (201)-829-5888. Email: [rjain@thumper.bellcore.com](mailto:rjain@thumper.bellcore.com)

additional information required to deliver or originate calls to those users, e.g. service profile information. A database at a higher level, i.e., an interior node, of the tree maintains information about the users registered in the set of RAs which are in its subtree. Thus, in Fig. 1, databases  $i$  and  $j$  serve RAs  $i$  and RA  $j$  respectively, and maintain information about PCS users registered in those RAs. The database which is the *least common ancestor* of  $i$  and  $j$  is denoted  $LCA(i, j)$ , and maintains information about users registered at all the leaves in its subtree, i.e., all the leaves shown in the figure. (Note that  $LCA(i, j)$  is not necessarily the root of the signalling network tree). The tree may be of arbitrary depth and width.

### 3 The basic user location strategy

The key features of the strategy used to locate mobile users in a tree-structured architecture can be described in terms of two operations [2]: A *FIND* operation, when a subscriber tries to place a call to a PCS user, and a *MOVE* operation when a PCS user crosses an RA boundary.

We call the *MOVE* operation without the use of caching a *BasicMOVE*, and it can be described by the following sequence of steps (see Fig. 1). The PCS user moves from the old RA,  $i$ , to RA  $j$  and registers at database  $j$ . A message is sent by database  $j$  to its parent (Message 1 in Fig. 1), which is queried to determine if a record for the PCS user exists there. If not, a record pointing to database  $j$  is created at the parent database. The message propagates up the tree (Message 2) until a database record is found for the user. This will occur at the database  $LCA(i, j)$ , and will consist of a pointer to a child database. The message is propagated down the tree following the downward chain of pointers (Messages 3 and 4) until the leaf database serving the user's old RA  $i$  is reached. The user is deregistered at the old database,  $i$ . An acknowledgement message is propagated up the tree (Messages 5 and 6), and deletes the downward pointers in the databases along its path, until it reaches  $LCA(i, j)$ . The acknowledgement message is propagated down the tree (Messages 7 and 8) until it reaches the new database,  $j$ .

We call the *FIND* operation without the use of caching a *BasicFIND*, and it can be described by the following sequence of steps. A caller at RA  $i$  places a call to a PCS user currently located at RA  $j$ . Since database  $i$  does not contain registration information for the called party, a message is propagated up the tree until a database which does contain information is found; this will be  $LCA(i, j)$ . A message is propagated down the tree from  $LCA(i, j)$ , following pointers until

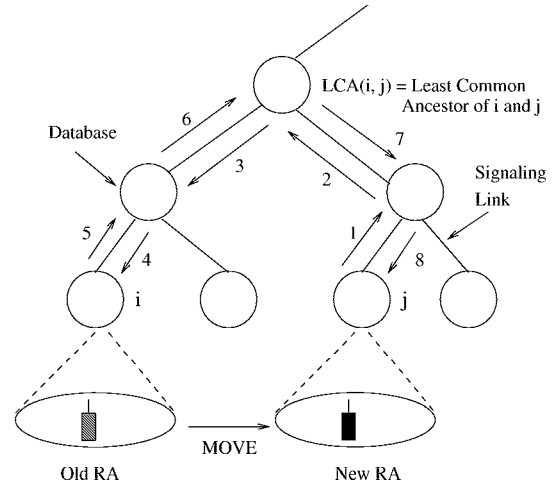


Figure 1: Steps in a *BasicMOVE* operation

the leaf database  $j$  is reached. Database  $j$  returns the information required to set up the call. This return message propagates up the tree until  $LCA(i, j)$ , after which it propagates down the tree until it returns to the caller's database  $i$ .

Minor variations of this basic user location strategy have been described by [13, 1].<sup>1</sup> If users tend to move among nearby RAs, and receive calls from nearby RAs, the *BasicFIND* and *BasicMOVE* operations will result in signalling messages being transmitted over relatively few links in the network, and the query or update of relatively few databases. However, two-level strategies such as those of IS-41 and GSM [8], require at most three databases to be queried or updated for similar operations (i.e., the HLR and the VLRs for two RAs). Thus the total number of database operations for each *FIND* or *MOVE* operation, and hence call setup time, may be greater in a tree architecture.

### 4 Caching user location

The basic idea behind caching is that, instead of traversing the tree from the leaves  $i$  and  $j$  to their ancestor  $LCA(i, j)$  for every call to a user, it may be possible to use the user's location from a previous call [7, 5].

We first describe one variation of caching, which we call *eager caching*. The *BasicFIND* operation is modified to become the *CacheFIND* operation (see Fig. 2). When a caller from RA  $i$  places a call to a PCS user at RA  $j$ , a message propagates up the

<sup>1</sup>The main difference is that in our description an explicit acknowledgement message is included as part of each operation, in a manner similar to IS-41 and GSM.

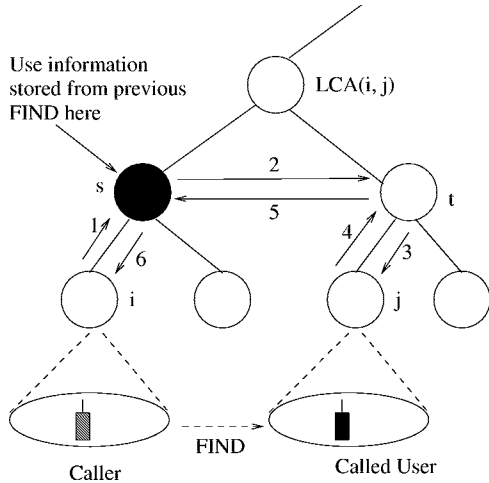


Figure 2: Steps in a *CacheFIND* operation

tree from  $i$  to  $LCA(i, j)$  and then down the tree to  $j$ , and an acknowledgement message returns from  $j$  to  $i$ . During the return path, however, a pair of *bypass pointers* is created. One bypass pointer is an entry at an ancestor of  $i$ , say database  $s$ , and points to an ancestor of  $j$ , say  $t$ , and is called a *forward bypass pointer*; the other pointer is from  $t$  to  $s$  and is called the *reverse bypass pointer*. (Note that  $s$  and  $t$  can be at different levels of the tree, although neither can be an ancestor of the other.)

During the next call to the same user from RA  $i$ , a message traverses up the tree until database  $s$  is reached (Message 1 in Fig. 2), and the forward bypass pointer to  $t$  is detected. The message travels to database  $t$  (Message 2), either via  $LCA(i, j)$  or via a shorter route, if it is available in the underlying CCS network. In either case, for the example in Fig. 2, the database at  $LCA(i, j)$  need not be consulted. Similarly, the acknowledgement message from the called party's database  $j$  will use the reverse bypass pointer at database  $t$  (Message 5) to avoid accessing the database at  $LCA(i, j)$ . In general, accesses to all the intermediate databases on the path from  $s$  to  $t$  via  $LCA(i, j)$  can be avoided.

In eager caching, the *MOVE* operation is modified to become the *EagerMOVE* operation as follows. When the user moves from RA  $j$  to a new RA  $k$ , a registration/deregistration message propagates from database  $k$  up the tree, via  $LCA(j, k)$ , to database  $j$ , as for a *BasicMOVE*. For example, suppose that a forward bypass pointer from database  $s$  to  $t$ , and a reverse bypass from  $t$  to  $s$ , have been created by a previous *CacheFIND*. Then as the registration message propagates from  $k$  to  $j$  via  $LCA(j, k)$ , several possibilities arise:

1. No bypass pointer is encountered. In that case,  $s$  and  $t$  are ancestors of  $LCA(j, k)$ , and their information is valid (i.e., the user is located in the subtree located at  $t$ ). No action other than that required for *BasicMOVE* is taken.
2. A reverse bypass pointer is found during the upward traversal of the registration message from  $k$  to  $LCA(j, k)$ . No action.
3. A forward bypass pointer is found during the upward traversal, or a (reverse or forward) bypass pointer is found during the downward traversal of the deregistration message (from  $LCA(j, k)$  to  $j$ .) In that case both  $s$  and  $t$  contain information which is no longer valid, and their bypass pointer entries are deleted.

It is clear that caching will only be beneficial for certain classes of users, those who receive calls from one or more subtrees frequently relative to the rate at which they cross RAs in different subtrees.

## 5 Costs and benefits of eager caching

In this section we present a preliminary analysis of the potential benefits of caching, using the eager caching strategy for illustration. Since the benefits depend upon numerous factors, we make some simplifying assumptions to obtain a first estimate.

We consider communication and database processing as our basic measures of cost. As a first step, let the cost of traversing any communication link be  $C$ , the cost of reading a database be  $R$  and updating a database be  $U$ . We also assume that the underlying CCS network is a tree and provides the only way for messages to be sent from one database to another.

Consider *FIND* operations which attempt to locate a particular PCS user. Refer to Fig. 2. If caching is used, the first *CacheFIND* results in a pair of bypass pointers being created, i.e., an increased cost of  $2U$  over *BasicFIND*. Subsequent *CacheFINDs* take advantage of these pointers to avoid accessing the databases along the path from  $s$  to  $t$  via  $LCA(s, t)$ . (Note that since the signalling network is a tree, the *CacheFIND* messages must still traverse the communication links). Let the path between  $s$  and  $LCA(s, t)$  be of length  $b$  hops and that between  $t$  and  $LCA(s, t)$  be  $b'$  hops. Then these subsequent *CacheFIND* messages enjoy a savings of  $2(b' + b - 1)R$  over *BasicFIND*. When the PCS user does move out of the subtree rooted at  $t$ , there is an increased cost of deleting bypass pointers of  $2U + (b' + b)C$ .

Let  $p(s, t)$  denote the average number of calls from the subtree rooted at  $s$  to the PCS user while it is in the subtree rooted at  $t$ . We refer to this quantity as the *Regional Call-to-Mobility Ratio (RCMR)* for a user with respect to tree nodes  $s$  and  $t$ .

Let the cost of a *MOVE* operation be denoted by  $M$ , the cost of a *BasicFIND* by  $F$ , and the cost of a *CacheFIND* by  $F'$ . Consider a *FIND* operation where the caller is at RA  $i$ , the called party is at RA  $j$  and  $LCA(i, j)$  is at  $r$  levels of the tree above  $i$  and  $j$ .

When a *BasicMOVE* occurs, and the user moves from RA  $j$  to RA  $k$ , the cost depends upon how far apart the two RAs are; suppose that  $LCA(j, k)$  is at  $q$  levels above  $j$  and  $k$ . For simplicity, in this example we assume that bypass pointers are always set up between the same levels of the tree, i.e.,  $b' = b$ .

Let the RCMR for this case be denoted  $p$ , the cost of a *CacheMOVE* by  $M'$ , the cost of the basic strategy be denoted by  $C_B$  and the cost of the caching strategy by  $C_C$ . Then  $C_B = M + pF$  and  $C_C = M' + pF'$ .

We will now consider two measures of performance in order to evaluate the benefits of caching. The ratio  $\frac{C_C}{C_B}$  denotes the relative total network cost of caching versus the basic strategy, and the ratio  $\frac{F'}{F}$  denotes the call setup time using caching versus the basic strategy. From the considerations above, we can show

$$\frac{C_C}{C_B} = 1 + X$$

$$X = \frac{4U + 2bC - (p-1)(2b-1)R}{(2q+1)U + 4(q+pr)C + (2q+1+p(2r+1))R}$$

$$\frac{F'}{F} = 1 + \frac{2U - (p-1)(2b-1)R}{p((2r+1)R + 4rC)}$$

These equations can be used to estimate the value of caching for different types of network costs and user calling and mobility patterns. As an example, we have plotted  $\frac{C_C}{C_B}$  for the case where  $r = 5$ ,  $q = 1$ , i.e., callers are from relatively remote locations compared to the called party, and the called party moves within a relatively localized region (see Fig. 3). In this case, we have only considered database impacts, and assumed that reading and updating a database have the same cost, i.e., we have set  $C = 0$  and  $U = R = 1$ . The ratio  $\frac{C_C}{C_B}$  is plotted for  $RCMR \geq 1$ , and is shown to decrease as RCMR increases, as expected. For a given value of RCMR, the curves show  $\frac{C_C}{C_B}$  for different values of  $b$ , i.e., when the bypass pointers are set up at different levels of the tree. It can be seen that for  $RCMR > 5$ , caching produces net cost reduction in this example, and the reduction can reach up to about 30%.

Similarly, Fig. 4 plots  $\frac{F'}{F}$  for  $RCMR > 1$  under the same assumptions, for various values of  $b$ . Again as

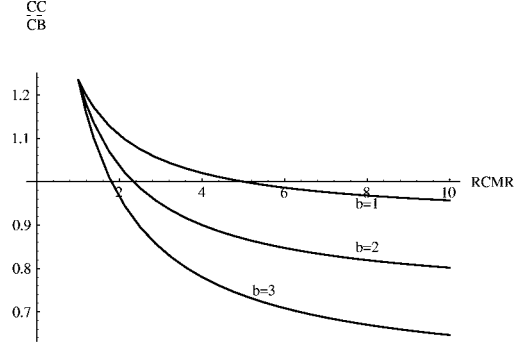


Figure 3: Relative total cost of caching and the basic strategy, for  $C = 0, U = R = 1, r = 5, q = 1$

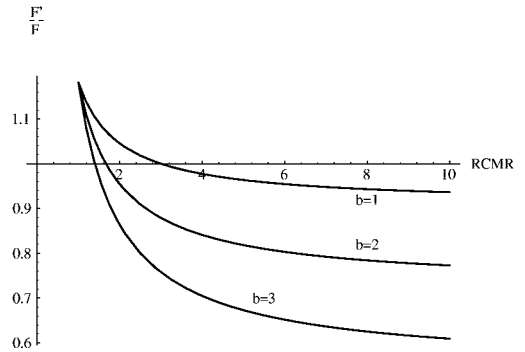


Figure 4: Relative call setup time of caching and the basic strategy, for  $C = 0, U = R = 1, r = 5, q = 1$

expected, the benefits of caching increase with RCMR, and caching can result in up to 35% reductions in call setup time in this example.

Note that unlike previous auxiliary strategies we have considered [7, 6], caching in this example results in a reduction of both total network cost as well as call setup time.

## 6 Discussion

We briefly discuss various issues we are currently addressing further.

**Lazy caching.** In lazy caching, the *MOVE* operation remains the same as *BasicMOVE*. If the user then moves from RA  $j$  to a new RA  $k$ , the *BasicMOVE* operation is performed (which does not access or modify any bypass pointers), and there are two possibilities for subsequent *CacheFIND* (See Fig. 5.) If the user moves to an RA which is within the subtree rooted at  $t$ , then the bypass pointers remain valid, and subsequent *CacheFIND* operations can continue

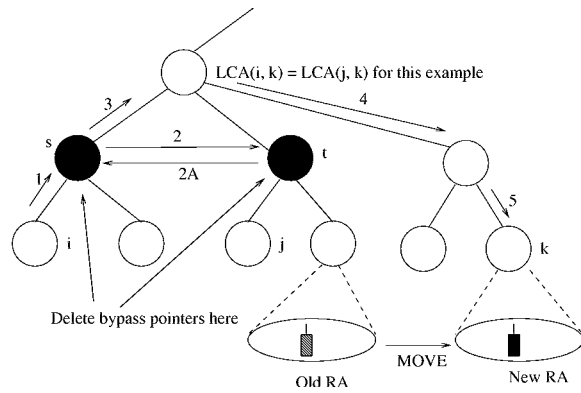


Figure 5: Steps when a lazy *CacheFIND* operation has a cache miss

to use them. If not, a subsequent *CacheFIND* operation will erroneously query database  $t$  (Message 2 in Fig. 5); this is called a *cache miss*. The reverse bypass pointer from  $t$  to  $s$  will be deleted and a failure code will be returned to database  $s$  (Message 2A); the latter will also delete its forward bypass pointer. The *CacheFIND* will then proceed as a *BasicFIND*, i.e., the message will propagate up the tree (Message 3) to  $LCA(i, k)$ , which will have a record for the called user.

Note that the caching strategy in [7] is a lazy caching strategy for a two-level tree.

**Bypass pointers at different levels.** This variation consists of choosing different levels for vertices  $s$  and  $t$ , i.e., levels at which bypass pointers are set up. In *simple* caching,  $s$  and  $t$  are both leaf vertices; this choice corresponds to implementing the caching scheme of [7] for tree networks (although note that in [7] a back-pointer is not set up). In *level* caching,  $s$  and  $t$  can be at any level, determined by the (long-term) call reception and mobility pattern of the called user, and in *adaptive* caching the levels can be set dynamically.

**Estimating user RCMR.** One possible scheme is not to estimate the RCMR on a per-user basis at all, but simply to apply a static form of caching (with fixed levels of bypass pointers, and eager or lazy variations) for certain parts of the network where user RCMRs are known to be high on average.

A more precise method is to estimate the RCMR on a per-user basis, by keeping a running tally of the number of calls received and number of RA crossings of a user, in the user's service profile. The RA from which the call originated (or to which a *MOVE* was made) is also recorded. To minimize storage used for this book-keeping, only tallies for a small number of RAs need be kept for each user, and these tallies can be processed to obtain the RCMR during off-peak hours.

Further work is needed to develop more efficient algorithms for RCMR estimation.

**Acknowledgement.** We thank Edward Lipper of Bellcore for comments on a draft of this paper.

## References

- [1] V. Ananthram, M. L. Honig, U. Madhow, and V. K. Wei. Optimization of a database hierarchy for mobility tracking in a personal communications network. In *Proc. Performance '93*, 1993.
- [2] B. Awerbuch and D. Peleg. Concurrent online tracking of mobile users. In *Proc. SIGCOMM Symp. Comm. Arch. Prot.*, Oct. 1991.
- [3] EIA/TIA. Cellular radiotelecommunications intersystem operations. Technical Report IS-41 (Revision B), EIA/TIA, July 1991.
- [4] C. Eynard, M. Lenti, A. Lombardo, O. Marengo, and S. Palazzo. A methodology for the performance evaluation of data query strategies in Universal Mobile Telecommunication Systems (UMTS). *IEEE J. Sel. Areas Comm.*, pages 893–907, Jun. 1995.
- [5] R. Jain. Reducing traffic impacts of PCS using hierarchical user location databases. In *Intl. Teletraffic Conf. Mini-Seminar on Mobility*, Montebello, Canada, Oct. 1994.
- [6] R. Jain and Y.-B. Lin. An auxiliary user location strategy employing forwarding pointers to reduce network impacts of PCS. *ACM Journal on Wireless Info. Networks*, 1995.
- [7] R. Jain, Y.-B. Lin, Charles Lo, and S. Mohan. A caching strategy to reduce network impacts of PCS. *IEEE J. Sel. Areas Comm.*, Oct. 1994.
- [8] R. Jain, Y.-B. Lin, and S. Mohan. Location strategies for personal communications services. In J. Gibson, editor, *Mobile Communications Handbook*. CRC Press, 1996.
- [9] C. N. Lo, R. S. Wolff, and R. C. Bernhardt. An estimate of network database transaction volume to support personal communications services. In *Proc. Intl. Conf. Univ. Pers. Comm.*, 1992.
- [10] A. D. Malyan, L. J. Ng, V. C. M. Leung, and R. W. Donaldson. Network architecture and signalling for wireless personal communications. *IEEE J. Sel. Areas Comm.*, pages 830–840, Aug. 1993.
- [11] K. Meier-Hellstern and E. Alonso. The use of SS7 and GSM to support high density personal communications. In *Proc. Intl. Conf. Comm.*, 1992.
- [12] Michel Mouly and M. B. Pautet. *The GSM System for Mobile Communications*. M. Mouly, 49 rue Louise Bruneau, Palaiseau, France, 1992.
- [13] J. Z. Wang. A fully distributed location registration strategy for universal personal communications. *IEEE J. Sel. Areas Comm.*, pages 850–860, Aug. 1993.