A Location Management Technique To Support Lifelong Numbering in Personal Communications Services

Derek Lam, Yingwei Cui, Donald C. Cox, Jennifer Widom
Electrical Engineering & Computer Science Depts.
Stanford University, Stanford, CA 94305
{dlam, dcox}@wireless.stanford.edu, {cyw, widom}@cs.stanford.edu

***Abstract*—This paper presents a novel *location management technique*, HOPPER, that is designed to support in a scalable and efficient manner non-geographical (lifelong) personal numbers in Personal Communications Services (PCS). Performance comparisons between our scheme and other schemes are derived from large scale simulations using a realistic traffic modeling framework for the ten largest cities of the United States. Results show that, in addition to inherently providing non-geographical numbers, the proposed scheme significantly improves lookup performance and requires relatively little database access and network signaling resources.**

I INTRODUCTION

A Personal Communications Services (PCS) [1] network maintains per-user information about subscribers in *user profiles*. A user profile contains information such as current location of the subscriber, authentication information, billing information, and other information related to advanced services such as call blocking, etc. User profiles are stored in *databases* maintained by service providers. Subscribers are located in system-defined geographical areas called *registration zones*. Databases are assigned to serve their zones and are connected to the wireline signaling network. When a PCS user makes a call, the network performs a *location lookup* to obtain from the callee's profile the current location of the callee and other information relevant to call delivery. When a user moves from one zone to another, the network performs a *location update*, modifying information in the profile to track the user's new location. We address the *location management* problem: efficient retrieval and maintenance of user profiles concentrating on location information. To focus the problem we assume that other information, e.g., for billing purposes, is handled by other parts of the system.

The current standards, IS-41 [4] and GSM[16], manage profiles in the databases using an *HLR/VLR* scheme. In this scheme, there are two types of databases: *Home Location Registers* (HLRs) and *Visiting Location Registers* (VLRs). Subscribers are assigned an HLR in a specific registration zone to permanently store their user profiles. VLRs are used to store profiles for subscribers currently visiting a zone other than their "home" zone. When a subscriber moves to a new zone, the profile in the subscriber's HLR must be updated. During a call setup, the network must retrieve the callee's profile. If the profile is not found in the caller's local VLR, the callee's HLR must be accessed. In IS-41 and GSM, each subscriber is assigned a phone number that contains enough information for the network to derive the location of the HLR. These phone numbers are, therefore, *geographical* and are tied to specific HLRs. For example, subscribers cannot maintain their same phone numbers if they switch service providers. Furthermore, if a subscriber moves permanently to a new region, he will either need to obtain a new number or will incur the cost of having his HLR at a zone usually remote from his current location [11].

In the future, PCS subscribers will be assigned *Non Geographical Phone Numbers (NGPNs)* that do not contain any information about their service provider or HLR. NGPN is an integral feature of PCS because it allows subscribers to have *lifelong* phone numbers. The main contributions in this paper are:

- We have developed a new *location management technique* (LMT), HOPPER, that uses a hierarchical organization of databases and on-line profile replication to achieve NGPN with scalability and efficiency.
- By performing extensive simulation studies, we have compared the performance of our proposal with other existing LMTs. Our dynamic performance simulations are based on realistic call/mobility models [11;13] and a topology modeled after the ten largest cities of the United States. As will be seen, our scheme performs very favorably against other schemes.

The rest of the paper is organized as follow. In Section II, we review related research in this area. We present our proposed LMT, HOPPER, in Section III. Section IV shows simulation results comparing the performance of HOPPER against other LMTs. Finally, Section V concludes the paper.

II RELATED WORK

There are a variety of ways to provide lifelong NGPNs. One approach is a *number translation* technique, which works in conjunction with the HLR/VLR scheme in the current standards. When the network needs to access an HLR, it first performs a lookup to translate the NGPN to the location of the subscriber's HLR. Once the location of the HLR is obtained, profile retrieval follows the conventional HLR/VLR protocol. Along these lines, [10] proposed using a *NGPN-to-HLR translation database* and presented a distributed hashing technique for maintaining the mapping between NGPN and HLR. [15] considered a technique that uses lookup tables and forwarding pointers to support NGPN. Our approach is to bypass number translation and directly search for profiles in the network. By organizing databases in an hierarchy and using replication techniques, we can support inherent NGPN without compromising scalability or efficiency. In Section IV, we show performance comparisons between our proposal and the number translation scheme.

Other LMT research focuses on efficiency in PCS and on decreasing profile lookup latency. Various techniques, such as local anchoring [6], forwarding pointers [8], and caching [9], have

Research partially supported by the National Science Foundation under grant# NCR-9628184 and Pacific Bell Corporation.

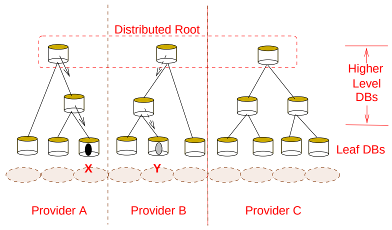

Fig. 1. Hierarchical Database Architecture

been considered to lower the signaling impact of PCS. Hierarchical database architectures [22] have been suggested to lower peak database access requirements. Off-line profile replication techniques [11;19] have been introduced to lower lookup latency given call and mobility statistics collected in advance. In this paper, we consider on-line replication—a natural evolution from off-line replication.

III HOPPER

In this section, we describe our proposed LMT, *Hierarchical Online Parametric ProfilE Replication (HOPPER)*. We first describe a simple hierarchical LMT with no replication and then present the details of our proposal.

III-A Basic Hierarchical Location Management

In a basic hierarchical LMT [22], the databases are organized in a multi-root tree hierarchy where the tree spanned by each of the roots belongs to one service provider (see Figure 1). Each *leaf database* serves a registration zone and stores profiles for subscribers currently located in that zone. Each *higher-level database* stores a pointer (subscriber ID + database ID) for every subscriber whose profile is currently stored in one of its descendent databases. The pointer is directed at the next lower level database that leads to the subscriber's current leaf database.

When X calls Y, the network first searches the local database at X's current zone for Y's profile. If this lookup fails, X's local database then propagates the query up the hierarchy to the first database that contains a pointer to Y's profile. The lookup query then propagates down the hierarchy following pointers until the leaf database with Y's profile is reached. In order to support multiple service providers, we require all root databases to cooperate and form a *distributed root* that contains pointers for all subscribers in the network. We assume that providers agree on a standard and allow each other to access their databases for user profiles. These assumptions are realistic given the current level of cooperation, e.g., in the European GSM cellular market.

When X moves from zone Z_i to zone Z_j, the PCS network must move X's profile to Z_j and delete the profile copy in Z_i. In addition, pointers in the higher-level databases spanned by the *least common ancestor (LCA)* of Z_i and Z_j, including the LCA itself, must be updated to reflect the new location of X.

We note that there is no concept of a "home database" for a subscriber as in HLR/VLR. Therefore, this architecture naturally supports lifelong NGPNs. The tradeoff is that in the hierarchical LMT location lookup may involve accessing several databases, thus potentially causing slow lookups and incuring high signaling loads. We now consider how to eliminate this disadvantage while maintaining NGPNs.

III-B On-Line Profile Replication

We consider *replication* as a way to improve performance in a hierarchical approach. The expense of replication is increased memory and profile update costs. Replication is different from *caching* [19] in that location information in replicated profiles is kept up-to-date. Also, in contrast to replication schemes in traditional distributed databases where expensive distributed locking [18] is needed, we only require updates to be propagated to replicas. An off-line replication technique was presented in [19] for faster lookup in HLR/VLR. Call/mobility statistics are collected for each user, and then based on estimated user *local call to mobility ratio* (LCMR) [8], replicas are selectively placed at databases while satisfying certain constraints, such as maximum number of replicas per subscriber. In [11], we presented a family of LMTs called HIPER, by extending the idea of off-line replication to a hierarchy of databases. It has been shown previously in [11;14;19] that there is sufficient locality in calling and mobility behaviors that replication is an efficient technique for improving performance.

In contrast to off-line schemes, our on-line replication technique uses on-line LCMR estimates and dynamically migrates replicas to appropriate databases. It offers the following advantages over off-line schemes.

- Off-line schemes estimate call/mobility traffic behavior and place replicas at locations to maximize expected benefits. On-line schemes can adapt to changes in user calling and mobility patterns by dynamically migrating replicas around the network.
- Due to practical storage constraints, it may not possible to place replicas at all locations where there are expected benefits. By dynamically adjusting the locations of replicas, on-line schemes can increase the "usefulness" of a replica and achieve the same lookup performance of off-line schemes with smaller storage requirements.

In the next subsections, we present HOPPER: a LMT that dynamically migrates replicas according to current traffic conditions. We show in Section IV through simulations that HOPPER attains a better dynamic lookup performance with smaller storage requirements than previous proposed off-line schemes.

III-C HOPPER Preliminaries

HOPPER's database architecture is based on the basic hierarchical scheme described in Section III-A. HOPPER attempts to place profile replicas at leaf databases where the cost of replication does not exceed its benefits. We use LCMR as the metric for evaluating the cost and benefit of replication. We define $LCMR_{i,j} = C_{i,j}/M_i$, where $C_{i,j}$ is the number of calls to user i originating from zone j, and M_i is the number of moves for user i in a given time period. HOPPER computes on-line estimates of $LCMR_{i,j}$ and migrates replicas to databases with sufficiently high LCMR values. HOPPER requires the network to collect and maintain per-user call and mobility statistics (i.e. $C_{i,j}$ and M_i), similar to other previously proposed off-line replication and caching schemes [8;11;19].

To support on-line replication, we make the following additions to the basic hierarchical scheme. Let $CPROF_i$ denote the profile copy of user i in his current zone, $RPROF_i$ denote all the profile replicas for user i, and $RPROF_i[k]$ to denote the k^{th} replica of user i. For brevity, we also refer to the leaf database for zone Z_j as DB_j and the database serving the current zone of user i as $DB_{i,curr}$.

1. In addition to the usual profile information, $CPROF_i$ maintains a list containing the addresses of all databases with $RPROF_i$.
2. A location lookup for user i originating from DB_j will result, if a profile is not found locally, in $DB_{i,curr}$ sending a copy of i's profile back to DB_j. We add a token bit to the profile message. If the token bit is set, DB_j will treat the profile as a replica. Otherwise, it will just obtain the relevant information from the profile for call setup and either delete or overwrite it afterwards.
3. When a user moves from an old zone to a new zone, in addition to the location update tasks in the basic hierarchical LMT, the old $DB_{i,curr}$ must also update the location information in all replica copies indicated in $CPROF_i$. We note that consistency can be maintained if the old $DB_{i,curr}$ first updates location information in its own copy and does not delete it until some guard period after all replicas have been updated.
4. Each DB_j maintains $C_{i,j}$, the number of calls for each user i originating from Z_j.
5. Each $CPROF_i$ and each $RPROF_i[k]$ stores M_i, the number of moves for each user i in a given time period. When a user moves to a new zone, the network must update location information in all profile copies; therefore M_i can easily be maintained at each profile copy by incrementing it during a location update.
6. $CPROF_i$ contains a "token counter" that stores the number of replica tokens for user i (see Section III-D.1).
7. Each database has an internal routing table that records its network hop distance from other databases in the network. It is expected that profile databases and network topology will not change very often. Thus, once the routing table is computed, it can be maintained in memory without much effort. The network hop distance is used in deciding where to move replicas (see Section III-E).

III-D HOPPER Parameterization

HOPPER is actually a family of algorithms based on its parameterization: HOPPER(N_{max}, R_{repl}, R_{del}, T_m, E_m, L) where N_{max} is the maximum number of replicas allowed for a user; R_{repl} and R_{del} determines whether a replica should be placed or deleted from a database, T_m and E_m define traffic statistics collection and on-line LCMR calculations, and L limits the amount of effort spent in on-line replication. We now discuss each of the parameters in detail.

III-D.1 Maximum Number of Replicas (N_{max})

N_{max} denotes the maximum number of replicas allowed for each user due to storage constraint considerations. We limit the number of replicas for user i by using a token counter method: $DB_{i,curr}$ is responsible for placing new replicas in the network and does that only when the token counter in $CPROF_i$ is not zero. The token counter in $CPROF_i$ is initialized to N_{max}. When a new replica is given out, the token counter is decremented. Similarly, if a replica is deleted from a database, $DB_{i,curr}$ must be notified and $CPROF_i$'s token counter will be incremented. We note that in HOPPER, replicas migrate between databases, but $CPROF_i$ controls the creation of new replicas.

III-D.2 On-line Replication Thresholds (R_{repl} , R_{del})

A common method to determine whether DB_j is a candidate for replicating i's profile is to consider $LCMR_{i,j}$. If $LCMR_{i,j}$ exceeds a certain threshold, R_{min}, DB_j is a candidate for replicating i's profile because the benefit of replication outweighs its costs. [5] proposes $R_{min} = 5$ for caching, and [19] suggests $R_{min} = 0.25$ for off-line replication in HLR/VLR. [11] proposes two thresholds: R_{min} and R_{max} for off-line replication. R_{min} is set in such a way that if $LCMR_{i,j}$ is below it, there is definitely no gain in replicating at DB_j. Similarly, R_{max} is set so that if $LCMR_{i,j}$ exceeds it then it is always advantageous to replicate at DB_j. [11] also discusses optimal values for R_{min} and R_{max} given off-line traffic statistics and the database topology.

Unfortunately, unlike off-line schemes, it is not possible to derive "optimal" thresholds for on-line techniques without more statistical knowledge of user traffic behavior. It is also expected that on-line optimal values will vary as calling/mobility traffic patterns change. We choose a somewhat ad-hoc approach and pick the same fixed thresholds for all subscribers. As will be seen, this approach does not diminish performance. We use two thresholds: R_{repl} and R_{del}.

- If $LCMR_{i,j}$ exceeds R_{repl}, then we try to move i's profile replica to DB_j, provided that the token counter in $CPROF_i$ is not zero. The details of our *Replica Distribution Protocol* are discussed in Section III-E.
- If DB_j currently has i's profile replica and its $LCMR_{i,j}$ falls below R_{del}, we delete i's replica from DB_j (see Section III-E.3).

Intuitively, if R_{repl} is set too high, it takes a large $LCMR_{i,j}$ value before the system attempts to move a replica to DB_j, thus degrading dynamic lookup performance. However, if R_{repl} is set too low, databases with low $LCMR$ will compete for replicas, possibly depriving or delaying databases with high $LCMR$ from obtaining replicas. If R_{del} is set too close to R_{repl}, a database can "ping-pong" between assigning and deleting replicas, thus wasting signaling bandwidth. Conversely, if R_{del} is set too low, replicas can remain at locations with minimal benefit and waste update cost. In the next paragraph, we discuss how we pick our thresholds. From our simulation results in Section IV, we see that the suboptimal thresholds chosen still provide excellent lookup performance under realistic time-varying calling and mobility traffic conditions.

Recall from Section III-A that in a hierarchical LMT, multiple databases may need to be accessed during location lookups. In deciding our on-line thresholds, we consider the total number of database accesses during location lookups and updates. Consider a user i who is currently served by $DB_{i,curr}$. We also consider DB_j with an on-line $LCMR_{i,j}$ estimate of R. We interpret R as the expected number of calls for user i originating from zone j for each move of i. We let b_l be the cost (including signaling messages) of one database lookup and b_u be the cost (including signaling messages) of one database update. We also let D be the hop distance between $DB_{i,curr}$ and DB_j along the database hierarchy. Suppose there is a replica of i in DB_j. Then:

- When there is a call to i from DB_j, DB_j can always retrieve i's profile locally resulting in a lookup cost of b_l.
- Each time user i moves, the system needs to update the replica at DB_j resulting in a cost of b_u.

Now consider when there is no replica of i in DB_j.

- When there is a call to i from DB_j, location lookup involves lookups at all databases between DB_j and $DB_{i,curr}$ along the database hierarchy, and the profile is finally retrieved from $DB_{i,curr}$. The total number of lookups is $D+1$ resulting in a lookup cost of $b_u * (D+1)$.
- Since there is no replica, there is no associated replica update cost.

Therefore, we can see that by replicating in DB_j, the gain in database lookup cost for each call from DB_j to i is:

$$b_l * D \tag{1}$$

However, we tradeoff b_u update cost for each move of i. It is then easy to show that R must satisfy the following condition before replicating at j is beneficial:

$$R > \frac{b_u}{b_l * D} \tag{2}$$

We observe that D is bounded by D_{min} and D_{max}, the minimum and maximum hop distances between two databases in the hierarchy. As a result, it is always advantageous to replicate when $R > b_u/(b_l * D_{min})$. Similarly, we should never replicate when $R < b_u/(b_l * D_{max})$. We choose our thresholds to be these bounds. Thus, $R_{repl} = b_u/(b_l * D_{min})$ and $R_{del} = b_u/(b_l * D_{max})$.

III-D.3 Traffic Statistics Collection (T_m, E_m)

In order to compute $LCMR_{i,j}$, we need to collect and maintain $C_{i,j}$ and M_i. Recall that these statistics are stored in the leaf databases and profile copies. For any on-line scheme, the collection method must have the ability to "forget the past", so current decisions are not based on long ago activities. We choose a simple method and clear $C_{i,j}$ and M_i (i.e. set them to zero) at a fixed interval defined by parameter T_m. To avoid inaccurate $LCMR_{i,j}$ immediately after zeroing, we only use $LCMR_{i,j}$ when the total number of call and move events exceeds a threshold, E_m, i.e., $C_{i,j} + M_i > E_m$.

III-D.4 Replica Search Distance (L)

When $LCMR_{i,j}$ exceeds R_{repl}, the system tries to migrate a replica to DB_j. L limits the amount of signaling spent in this process. The use of L is discussed further in the Replica Distribution Protocol, covered next.

III-E Replica Distribution Protocol

For our Replication Distribution Protocol, we first describe the general protocol in Section III-E.1. In Section III-E.2, we discuss how replicas are exchanged between databases and in Section III-E.3, we cover deletions of replicas.

III-E.1 General Protocol

Our replica distribution protocol proceeds as follows. Suppose we have a call to user i from DB_j and there is no replica of i in DB_j. Location lookup will find $CPROF_i$ at $DB_{i,curr}$. Recall that location lookup is initiated from DB_j. In HOPPER, the following events occur.

1. Initially, DB_j sends $C_{i,j}$ along with the normal lookup message to its parent database. Eventually, the message will be received by $DB_{i,curr}$ where $CPROF_i$ is found.
2. Before $DB_{i,curr}$ sends i's profile to DB_j, it first calculates $LCMR_{i,j}$ using $C_{i,j}$ in the lookup message and M_i maintained in $CPROF_i$. There are three cases to consider:
   - **Case A:** If $LCMR_{i,j} > R_{repl}$ and $CPROF_i$'s token counter is not zero, then $DB_{i,curr}$ decrements the token counter and sends a profile to DB_j with the token bit set to one.
   - **Case B:** If $LCMR_{i,j} > R_{repl}$ but $CPROF_i$'s token counter is zero, then $DB_{i,curr}$ returns a profile with the token bit set to zero to DB_j. $DB_{i,curr}$ then executes the *Replica Exchange Protocol* (Section III-E.2).
   - **Case C:** If $LCMR_{i,j} <= R_{repl}$, then $DB_{i,curr}$ simply sends a profile to DB_j with the token bit set to zero.
3. DB_j receives the profile from $DB_{i,curr}$. There are two cases to consider:
   - **Case A:** DB_j examines the token bit and finds that it is set to one. DB_j will store the profile as a replica and acknowledge $DB_{i,curr}$. When $DB_{i,curr}$ receives the acknowledgment, it will update its replica address list.
   - **Case B:** DB_j finds a zero token bit. DB_j will not treat the profile as a replica and will move on to its other tasks in location lookup.

III-E.2 Replica Exchange Protocol

We now describe the protocol for migrating user i's replica from one leaf database to another when the system has already distributed all N_{max} replicas of i, but finds a new database with $LCMR_{i,j} > R_{repl}$ (Event 2B in Section III-E.1). We emphasize that the replica exchange protocol happens asynchronously with the normal location lookup procedure. As a result, replica exchanges do not delay location lookups. These events are "spawned" from Event 2B in Section III-E.1.

1. After returning the profile to DB_j, $DB_{i,curr}$ randomly picks a database, say DB_r, from among its list of current replica addresses. If the hop distance, $d_{curr,r}$, between $DB_{i,curr}$ and DB_r, is smaller than L where L is the replica search distance parameter described previously, then $DB_{i,curr}$ sends $C_{i,j}$ (obtained from DB_j previously) to DB_r to consider "returning" its replica. If $d_{curr,r} > L$, no further event will

occur. Note that the selection method is relatively simple. We discuss later in this subsection why we choose it and a possible alternate method.
2. DB_r receives the message from $DB_{i,curr}$. It compares the expected savings in database lookups between replicating at DB_j and DB_r by looking at the quantities $C_{i,j} * D_{curr,j}$ and $C_{i,r} * D_{curr,r}$ (see Equation 1). If $(C_{i,j} * D_{curr_j}) > (C_{i,r} * D_{curr,r})$, then DB_r deletes its replica of i and sends a message to inform $DB_{i,curr}$. Otherwise, no further events occur.
3. If $DB_{i,curr}$ receives notification that DB_r has deleted its replica, it then sends a profile copy to j with token bit set to one.
4. When DB_j receives the profile copy, it first acknowledges $DB_{i,curr}$ and stores the copy as a replica.

Since initially HOPPER gives out replicas in a first-come-first-serve manner, databases with relatively low $LCMR$ values can deprive databases with higher $LCMR$ values from obtaining replicas. Therefore, our replica exchange protocol attempts to migrate replicas to databases with higher $LCMR$ values to improve performance. $DB_{i,curr}$ randomly chooses a database already containing a replica in Event 1 to involve all replica databases in the migration process, as opposed to always picking the closest database. Because of random database selection, L serves to bound the maximum signaling per replica exchange. A more sophisticated selection method is to consider the entire replica address list in Event 1 and randomly pick a database from among those that satisfy the maximum signaling bound, L. However, we observe from our simulation results in Section IV that the simple method performs very well. Thus, we choose it for its lower complexity. Also, $CPROF_i$ has control over how replicas get exchanged. During high traffic situations, e.g. when there are excessive location lookups from many different databases for user i, $CPROF_i$ can stop replica exchange altogether.

III-E.3 Deletion of Replicas

At each location update for user i, the old $DB_{i,curr}$ updates the location information in each replica. It waits for acknowledgements from all replica databases before forwarding $CPROF_i$ to the new $DB_{i,curr}$. When a replica database, DB_r, receives an update message, it updates its location information and increments its value of M_i. It also obtains $C_{i,j}$ locally to form $LCMR_{i,j}$. If $LCMR_{i,j}$ falls below R_{del}, then DB_r deletes the replica and notifies $DB_{i,curr}$ in its acknowledgement message. The old $DB_{i,curr}$ will increment $CPROF_i$'s token counter and update the replica address list before forwarding $CPROF_i$ to the new $DB_{i,curr}$.

IV SIMULATIONS

IV-A Simulator and Simulation Topology

We have studied the performance of various LMTs, including HOPPER, using our flexible discrete-event simulator [11;14], *Pleiades*. We performed our simulations on a topology modeled after the largest ten cities of the United States. Figure 2 shows our simulation geography and network signaling topology and Figure 3 shows our database topology where we consider a 4-level database hierarchy. Four databases form a distributed root (recall Section III-A), storing pointers for all subscribers. The number of databases assigned for each city is proportional to its reported population in [21] with each one serving approximately one million subscribers. The number of registration zones is also proportional to the reported sizes of the cities. Each registration zone is approximately 400 square miles in area.

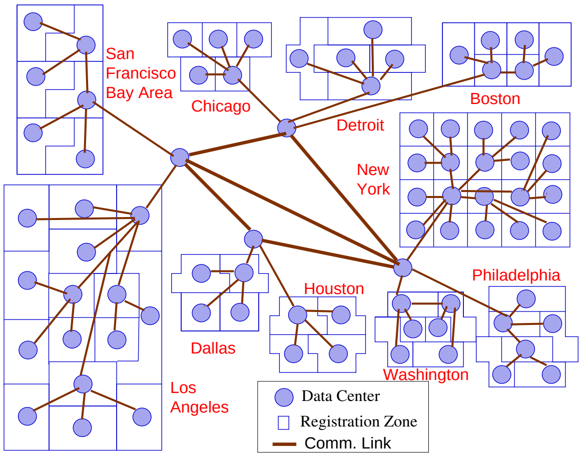

Fig. 2. U.S. Simulation Geography and Signaling Topology

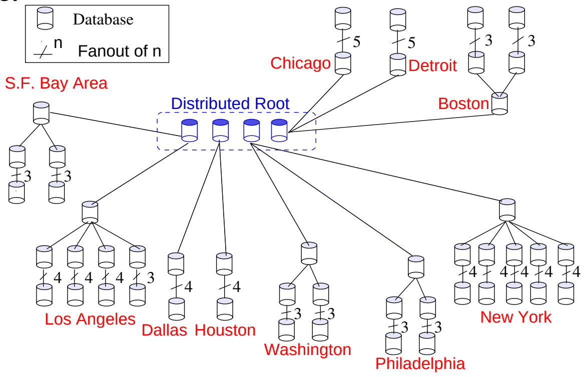

Fig. 3. Database Hierarchy

IV-B Dynamic Performance Simulation Results

For the following simulations, our goal is to investigate the performance of various LMTs under realistic traffic conditions. By using time-varying traffic models, we are able to study transient and peak performance of the LMTs.

Our call and mobility models have been discussed in [13;14]. The call model includes various call traffic types and models call volume changes throughout the day. It also incorporates callee distribution by maintaining, for each caller, a list of the people they call most often with the probabilities of making calls to each of them. We have found empirically that the callee probability distribution can be modeled accurately by using *Zipf's Law* [14]. We have instantiated our call models with parameters we derived from call traffic data [20] from our university telephone exchange. We used the Metropolitan Mobility Model (METMOD) and the National Mobility Model (NATMOD) from [13] for modeling movements within and between the cities, respectively. These mobility models include various movement types and account for traffic volume changes over time. They are derived from vehicle traffic data [17], transportation surveys [3;7;12], and commercial airline data [2]. We now present simulation results comparing the performance of the following LMTs.

- **Caching HLR/VLR with NGPN translation.** We simulate an HLR/VLR scheme with caching [8] for $R_{min} = 5$. We augment this scheme to support NGPN by using an idealized model of the number translation technique (recall Section II). We assume that during location lookup, a NGPN-to-HLR translation is required only when a copy of the profile is not found at the local database. We further assume that at every VLR, there is a local translation database. As a result, each NGPN-to-HLR translation incurs only one lookup and no network signaling message. These assumptions correspond to the best possible scenario for the number translation scheme, and we ignore all updates and signaling activities associated with maintaining such a translation database at every VLR in the network.
- **HIPER.** We simulate off-line replication with HIPER(N,L) [11]. We set $N = 5$ to allow a maximum of five replicas per subscriber. We set $L = 3$ to allow replicas to be placed anywhere in our 4-level database hierarchy. This set of parameter values corresponds to the best lookup performance and results in the lowest database access and signaling loads for HIPER.
- **HOPPER.** We simulate HOPPER with $N = 5$ and $b_u = b_l = 1$. We set $R_{repl} = 0.5$ and $R_{del} = 0.125$ because $D_{min} = 2$ and $D_{max} = 8$ in our database topology, respectively. Also, $T_m = 24$ hrs, $E_m = 2$, and $L = 8$. L is set so that we are not restricting how we migrate replicas between databases.

We simulated 5,000,000 PCS subscribers, corresponding to approximately 6% of the total population of the cities. We have performed a multi-day simulation and are showing results for a representative 24 hour period because there is no significant difference between the results from different days, after start-up transients. Figure 4 shows the *local lookup percentage* versus simulation time. Local lookup percentage is defined as the percentage of all profile lookups that are satisfied at the local database. As expected, HOPPER attains the best lookup performance, averaging over 90% throughout the 24 hour period. Table 1 shows the storage requirements for all simulated schemes normalized to the

Scheme	Memory Requirements
Caching HLR/VLR	1.0
HIPER	1.46
HOPPER	1.08

Table 1. Relative Memory Requirements

storage needed by the Caching HLR/VLR scheme. We have only quoted the memory requirements for storing user profiles and ignored the storage requirements for the translation database. Since there is a translation database at every VLR, the actual memory

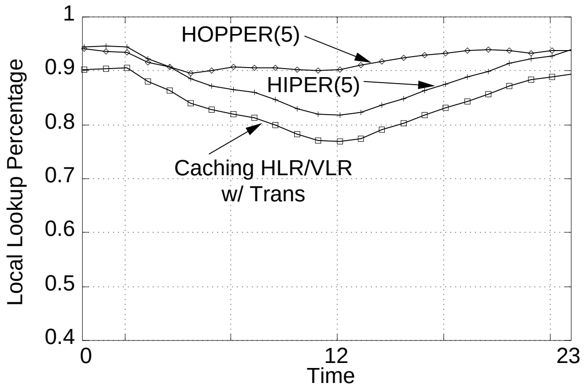

Fig. 4. Local Lookup Percentage

requirement could be significantly higher. Even with this pessimistic assumption, HOPPER is still competitive having memory requirements not much more than caching HLR/VLR. One interesting fact here is that by dynamically reallocating and deleting replicas, HOPPER is able to attain its improved lookup performance with a smaller memory requirement than off-line HIPER.

Figure 5 shows the total number of database accesses required by the different schemes. For all the schemes, we quote the total database activities from all subscribers' calling and mobility activities. For HOPPER, we have included the database activities introduced by on-line replication. For Caching HLR/VLR, we have included NGPN translation lookups. However, for HIPER, we did not include replication activities because they are expected to be done off-line. We see from Figure 5 that HOPPER incurs the smallest amount of database access, even after including on-line activities. HOPPER performs significantly better than Caching HLR/VLR in this category.

Finally, in Figure 6, we show the amount of network activity in terms of the global number of message-hops. In this category, we see that Caching HLR/VLR introduces the lowest signaling load. However, the absolute values between all four schemes are similar.

V CONCLUSIONS

In this paper we have presented an efficient location management technique, HOPPER, that supports lifelong Non Geographical Personal Numbers (NGPN). HOPPER supports NGPN by using a hierarchy of databases, and it achieves better lookup performance than previously proposed schemes by using an on-line user profile replication technique. We have performed extensive simulations on a model of the ten largest cities of the United States; we show that HOPPER achieves excellent lookup performance and has low database access requirements while incuring little signaling cost. Simulation results also show that HOPPER requires little, if any, additional memory comparing to the Caching HLR/VLR scheme. Thus, HOPPER is a competitive LMT for providing lifelong numbering given its excellent dynamic lookup performance and low database access requirements.

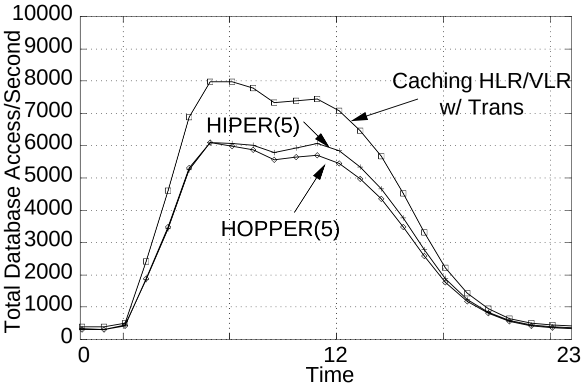

Fig. 5. Global Database Access Rate

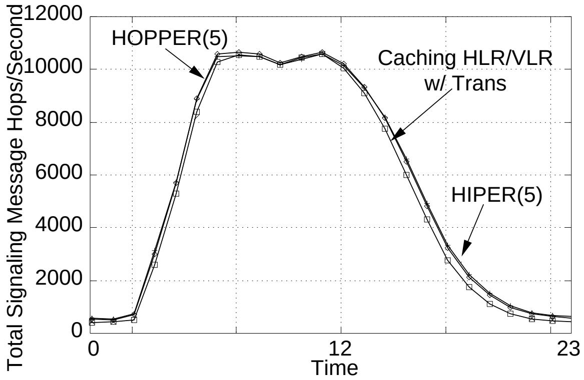

Fig. 6. Global Signaling Hops

REFERENCES

[1] D. C. Cox. Wireless personal communications: What is it? *IEEE Personal Communications*, pages 20–35, April 1995.

[2] Origin and destination survey, t-100 domestic market data. Department of Transportation. Dataset for periods 10/91-12/91 and 10/93-12/93.

[3] ECMT. *European Transport Trends and Infrastructural Needs*. OECD publications, 2, Rue Andre-Pascal, 75775 Paris CEDEX 16, France, 1995.

[4] EIA/TIA IS-41.3 (Revision B). *Cellular Radiotelecommunications Intersystem Operations*, July 1991.

[5] H. Harjono, R. Jain, and S. Mohan. Analysis and simulation of a cache–based auxiliary location strategy for PCS. In *IEEE Conference on Networks and Personal Communications*, 1994.

[6] J.S.M. Ho and I.F. Akyildiz. Local anchor scheme for reducing location tracking costs in PCN. In *1st ACM International Conference on Mobile Computing and Networking (MOBICOM'95)*, pages 170–180, Berkeley, California, 1995.

[7] P. S. Hu and J. Young. *1990 NPTS Databook: Nationwide Personal Transportation Survey*. Federal Highway Administration, November 1993.

[8] R. Jain and Y. Lin. An auxiliary user location strategy employing forwarding pointers to reduce network impacts of PCS. In *International Conference on Communications, ICC '95*. IEEE, 1995.

[9] R. Jain, Y.-B. Lin, 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] R. Jain, S. Rajagopalan, and L. F. Chang. Phone number portability for PCS systems with ATM backbones using distributed dynamic hashing. *IEEE Journal of Selected Areas in Communications*, 15:96–105, January 97.

[11] J. Jannink, D. Lam, N. Shivakumar, J. Widom, and D. C. Cox. Efficient and flexible location management techniques for wireless communication systems. In *2nd ACM International Conference on Mobile Computing and Networking (MOBICOM'96)*, pages 38–49, Rye, New York, November 1996.

[12] R. Kitamura. *Time-of-Day Characteristics of Travel: An Analysis of 1990 NPTS Data*, chapter 4. Federal Highway Administration, February 1995.

[13] D. Lam, D. C. Cox, and J. Widom. Teletraffic modeling for Personal Communications Services. *IEEE Communications Magazine Special Issue on Teletraffic Modeling, Engineering and Management in Wireless and Broadband Networks*, 35(2):79–87, February 1997.

[14] D. Lam, J. Jannink, D. C. Cox, and J. Widom. Modeling location management for Personal Communication Services. In *Proceedings of 1996 IEEE International Conference on Universal Personal Communications (ICUPC96)*, pages 596–601, September 1996. ftp://wireless.stanford.edu/ derek/.

[15] Y.-B. Lin. A cache approach for supporting life-time UPT number. *Wireless Networks*, 2:155–160, 96.

[16] M. Mouly and M.-B. Pautet. *The GSM System for Mobile Communications*. Palaiseau, France, 1992.

[17] 1990 commute summary. Metropolitan Transportation Commission, Lotus 123 format spreadsheet, August 1990. S.F. Bay area transportation measurements.

[18] M. T. Oszu and Valduriez P. *Principles of Distributed Database Systems*. Prentice Hall, 1991.

[19] N. Shivakumar and J. Widom. User profile replication for faster lookup in mobile environments. In *1st ACM International Conference on Mobile Computing and Networking (MOBICOM'95)*, pages 161–169, Berkeley, California, 1995.

[20] Telephone call traffic data set, 3/95–9/95. S. Phillips, personal communication, October 1995. Encrypted ID numbers.

[21] International population data. U.S. Bureau of Census, International Data Base.

[22] 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.