Data Management for User Profiles in Wireless Communications Systems*

Jan Jannink, Derek Lam, Narayanan Shivakumar, Jennifer Widom, Donald C. Cox

Computer Science & Electrical Engineering Depts.
Stanford University, Stanford, CA 94305

{jan, shiva, widom}@db.stanford.edu, {dlam, dcox}@wireless.stanford.edu

Abstract

The explosive growth in wireless communications systems and the demand for advanced mobility features have created novel data management problems. Current schemes to address these problems rely on database organizations that have limited functionality and performance anomalies. We propose a new data management scheme that is flexible and scalable, and that is incrementally deployable so it can coexist with current data management standards. We compare our protocol against current standards and other suggested protocols using realistic calling and mobility patterns. To do so, we have built *Pleiades*, an extensible event driven simulator that is easily configurable to arbitrary geographies, networks, calling and mobility patterns, and data management schemes. We present, for the first time, models to closely approximate user calling and mobility patterns. These models are validated against real call traffic and urban vehicle data. Based on simulations for a representative 24-hour period in the San Francisco Bay Area, we report on several performance measures: global and local database transaction load in terms of number of reads and writes, and network activity in terms of number of messages and average number of message hops.

1 Introduction

In a wireless communication system, mobile users place and receive calls through a wireless medium, but for efficiency reasons as much communication as possible takes place on a wireline network. When a user places a call, the wireless communication infrastructure must perform several tasks including authenticating the caller, locating the callee, routing the call to the callee, and updating billing records. Databases constitute a critical component in supporting all the above functionalities in any wireless infrastructure, making it imperative to deploy the right database technology and data management protocols for a wide area wireless infrastructure.

The novel requirements for databases in traditional telephone communications systems have been recognized as a challenging problem [GR93, Jag94]; the additional data management factors associated with mobile users only make the problem more complex. In this paper we consider how to efficiently manage *user profiles*, which contain data such as user location information and system billing information, in a nationwide wireless infrastructure. We consider how to perform user

*Research partially supported by the Center for Telecommunications and the Center for Integrated Systems at Stanford University, by the Anderson Faculty Scholar Fund, and by equipment grants from Digital and IBM Corporations.

profile lookups for call setup and profile updates when users move. We include cost effectiveness and backwards compatibility as important dimensions of data management in wireless networks.

Any proposed solution to such data management problems needs to be scalable due to the explosive growth in number of wireless customers [WK95] and the trend towards smaller registration areas. Also, since call setup needs to occur within 170 milliseconds of call initiation [HP90], the database accesses and network signaling required by the data management mechanisms must have fast response times. Due to the high volume of calls and high user mobility, data management schemes should also have high throughput. In addition, features such as lifelong *location independent* numbering are attractive to network subscribers but impose additional constraints on the data management techniques. All the above requirements need to be handled simultaneously in any useful nationwide wireless infrastructure, thereby making it important to understand the tradeoffs of the many variations of profile management techniques.

In this paper we consider various approaches to structuring the system data. We first study the two current wireless telephony standards, *IS-41* and *GSM*, and other suggested database structuring mechanisms. We explain the performance anomalies and limitations of these schemes. We then propose a new hierarchical organization of autonomous databases that provides flexible location independent numbering. Since some of the standards for data management are already functioning in current systems, one of the key features of our proposed scheme is that it is designed to be incrementally deployable in a nationwide network: our scheme can coexist with current limited data management schemes while the databases evolve to our more flexible approach. To study the performance penalties of our scheme, we evaluate it through a comprehensive simulation study along with the current standards and other "optimal," but impractical, profile management schemes.

We present the architecture of *Pleiades*, an extensible event-driven simulator we have built that can be configured to handle arbitrary data management schemes and simulate wide-area networks and large user populations—beyond 3 million. Using Pleiades we model calling and mobility patterns of users that have been validated against several months of real call data and urban vehicle traffic data, an aspect we have not seen in other simulation work. Another unique aspect of our work is that we use the actual geography of the San Francisco Bay Area to simulate the different protocols over a representative 24-hour period. From our simulations we report on performance measures that are critical in evaluating the various protocols: We present the number of database profile operations per second in terms of global (all databases) and local (a sample database) read and write operation loads, along with the storage space required for the user profiles. We also evaluate the network activity in terms of number of messages—characterizing bandwidth—and average number of messages per hop—characterizing latency. We see from the results that our new profile management scheme performs close to current standards and better for some performance measures, while enabling location independent numbering.

The rest of the paper is structured as follows. In Section 2 we outline the data that needs to be stored by the wireless system. We then present the architecture of a generic wireless system infrastructure and explain the messages that need to be handled by databases. These messages are used to evaluate the data management techniques described in Sections 3 and 4. In Section 3, we concentrate on the current techniques used in small wireless systems and show why we expect them not to scale for wide-area wireless infrastructures. In Section 4, we describe our new hierarchical data management scheme, along with other schemes we use for measurements. In Section 5, we describe the architecture of Pleiades, our extensible simulator and present our user calling and mobility models. In Section 6, we discuss the telephone call and vehicle traffic data with which we derived our user models. In Section 7, we report our empirical results that contrast the different data management schemes along with our observations on the results, and we conclude with Section 8.

1.1 Related Work

Previous surveys of data management techniques for wireless networks [IB93, Wan94] have not provided quantitative comparisons. In this paper we evaluate the techniques, including our new scheme, by comparing both the database and network aspects of the various schemes. Some other work provides simulation results [KVP94, IB92, PMG95] similar to ours, but this work relies on assumptions that incompletely characterize actual human behavior and physical geographies. We present, for the first time, accurate models characterizing calling and mobility patterns of users that have been validated against several months of actual calling data and actual traffic patterns in the San Francisco Bay Area. In addition, our event-driven simulator scales to multi-million user population sets and we provide comparisons of the protocols for a representative 24-hour period to help us observe the variations in database and network loads during different times of the day. In this paper, we do not consider caching [HJM94, JLLM94] or off-line user profile replication [SW95], since such schemes are complementary to the underlying data management techniques and thereby do not contribute towards evaluating the different ways of organizing the data.

2 Wireless Infrastructure

In this section, we first describe the *user profile* that needs to be maintained for every wireless service subscriber to properly operate the wireless network. We then present the architecture of a generic wireless infrastructure.

2.1 The User Profile

The infrastructure maintains the following information for each of its subscribers in some set of databases.

1. *System Information:* The system needs to maintain information (in some database or set of databases) about where the user is located at any point in time. The system also needs to maintain authorization information (such as passwords, or personal identification numbers)

for reasons of security and user privacy. Current call status, such as BUSY, ACTIVE, or INACTIVE, also fits in this category.

2. *Service Provider Information:* The system usually maintains logs of each user's calls as billing records. It may also maintain mobility patterns of users for predictive caching and replication [SW95].

3. *Feature Support Information:* The system may maintain information needed for "value-added" services such as numbers for call screening or call blocking.

Note that each of the above classes of information may be vertically partitioned or replicated across multiple databases for performance reasons. The data partitioning is beyond the scope of this paper, but we shall see later how the different profile management schemes that we evaluate replicate the user profiles.

In this paper, we focus primarily on the system information, since it is continuously queried, due to calls, and updated, due to user moves. These two events have the greatest impact on the database load as well as on the load of the underlying wireline network that ships the queries and updates.

2.2 The Network Infrastructure

In this section, we first describe the architecture of a generic wireless system and show how databases fit into the infrastructure. We then describe the events we deal with that occur in the wireless systems and need to be serviced by database queries or updates.

Figure 1 represents a canonical wireless network architecture. Mobile users converse with the radio port closest to their location through the wireless medium. Several radio ports are serviced by a *Mobile Switching Center* (MSC) that is connected to other MSCs through a wireline network. Each MSC corresponds to a *registration area.* We also see in the figure several databases that are connected to the wireline network. These databases are used to store the user profile information described in Section 2.1. When a call is initiated from a user A to a user B, the MSC of user A causes a query to be sent to some database(s) on the network in order to find the location information for B (i.e., B's registration area). Then the call is setup from user A to B (shown using dark lines in the figure) through the MSC of A, to the MSC of B through the wireline network, then to the radio port connected to the MSC of B, and finally over the wireless link to B. As mentioned earlier, user profile information may be replicated or partitioned across the databases for performance reasons. For example, in the figure we see A's profile replicated at several databases.

We now outline the principal events that generate database and network traffic. We describe the following using the abstract operations LOOKUP-FIRST and UPDATE-ALL. LOOKUP-FIRST finds the first database close to a given MSC that contains a copy of a given user's profile information. UPDATE-ALL updates all databases that contain copies of a given user's location information.

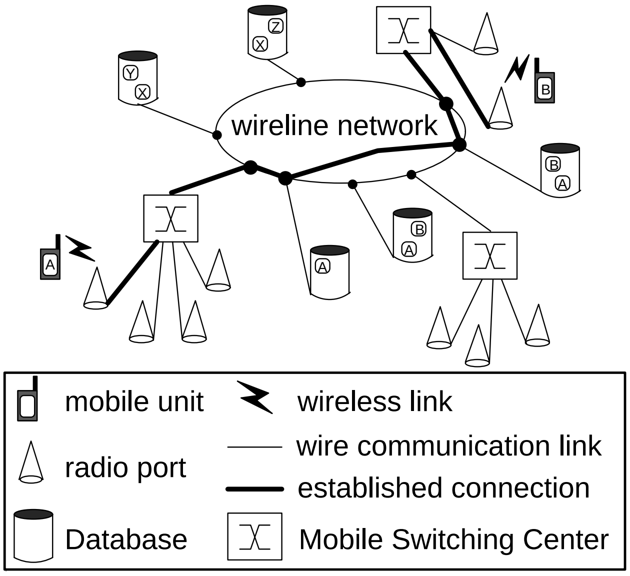

Figure 1: Wireless Network Structure

- **Calls:** When a user calls another network subscriber, a call setup procedure is initiated. The caller's MSC uses LOOKUP-FIRST to find a database that has a copy of the callee's location information, queries the database, and then sets up a network connection between the users. A billing database is also updated to maintain a log of the user's call.

- **User Move:** When a user moves between registration areas, he registers with his new MSC and deregisters from the previous MSC. The system also performs an UPDATE-ALL.

- **Switch On:** When a user switches his communication unit on, the unit sends a message to the closest radio port to initiate registration. The MSC of the radio port queries the LOOKUP-FIRST database for that user's authorization information, and also possibly creates replicas of the user's profile at various databases.

- **Switch Off:** When a user turns off his communication unit, the unit causes an explicit deregistration of the user from his MSC, and also invalidates any replicas of the user profile on databases that store replicas.

In subsequent sections, we show how LOOKUP-FIRST and UPDATE-ALL vary for the different data management schemes we study in the paper. For completeness, pseudocode for the MSC functions is given in the full technical report [JLWC95].

3 Current Data Management Standards

We first outline the two wireless telephony standards and the common way of structuring databases in those standards. We then present some anomalies in the standards that make them impractical for nationwide or global use. We also show some of the drawbacks that make the standards inconvenient and inflexible from the user's perspective.

3.1 Current Standards

There are two parallel standards in wireless telephony today, *IS-41* [EIA91] in the United States and *GSM* [MP92] in Europe. For the purposes of this paper, the two standards are the same, although GSM uses more sophisticated authorization mechanisms. We first outline how user location lookups take place during call setup and movement updates are performed as part of the handoff procedure, in order to understand precisely how they make use of databases. The details of these processes are critical, since they affect the read and write loads on the databases that store user profiles. We then consider how the databases themselves are structured.

Figures 2 and 3 show simplified pseudocode for the lookup and update algorithms in the standards. This pseudocode shows the operations taking place at a single database to access profile information. Notes and definitions as well as more detailed code can be found in [JLWC95]. Both standards rely on what is commonly known as the *HLR/VLR* scheme: Each user, when subscribing to the network, gets a statically assigned *Home Location Register* or HLR database that always maintains location information for the user.

When the user travels to a new registration zone, his HLR is updated and an entry in the *Visitor Location Register*, VLR database, of his new zone is created for him. The actual sequence of messages is shown in Figure 5. We see that a user who moves into a new zone first registers with the local VLR, which then updates the user's HLR. The user's HLR invalidates the profile in the VLR of the user's previous zone, and then forwards a copy of the user profile to the new VLR.

When a user wants to make a call to another user, the caller's MSC first checks the local VLR for the callee's profile (a message not specified in the signaling standards, as the VLR and MSC are assumed to be located together) If the VLR doesn't contain the callee's profile, a message goes to the callee's HLR (we discuss below how to compute the callee's HLR) querying the callee's location. The callee's HLR verifies the callee's VLR, returns it to the MSC which then initiates the connection between the callee and the caller. Again, the sequence of messages is shown in Figure 4. In the terminology of Section 2.2, LOOKUP-FIRST is first the VLR, and on a VLR miss, the HLR of the callee. UPDATE-ALL updates the HLR and the relevant VLRs of the user who moved.

We now consider how the user profile information is organized in the various databases. Unfortunately, the standards do not specify a relationship between geographical entities (such as MSCs) and database entities (HLRs and VLRs). In practice, as assumed in the standards, each MSC may use its own local database for a VLR. HLRs, corresponding to a home location for a subset of users, may also be found at the location of an MSC. HLR and VLR databases may be combined into a single database.

Lastly, consider how an MSC finds the HLR of a callee. One simple approach would be to store at MSCs an additional database that maps caller IDs to HLR database locations. This approach

```
Loc Lookup(calleeID) {
  loc = Query(calleeID)
  home = HomeDB(calleeID)
  if (thisDB is home)
    return loc->Certify(calleeID)
  else if (loc is null)
    return home->Lookup(calleeID)
  else
    return loc
 }
```

Figure 2: HLR/VLR Profile Lookup

```
Update(userID, thisMSC) {
  Update(userID, thisMSC)
  home = HomeDB(userID)
  if (thisDB is home) {
    oldDB = OldDB(userID)
    oldDB->Cancel(userID)
  }
  else
    home->Update(userID, thisMSC)
 }
```

Figure 3: HLR/VLR Profile Update

requires additional storage at every site, and has maintenance problems since an entry must be added to every MSC for every subscriber. In practice, systems perform a static partitioning of numbers so that the caller ID indicates the location of the HLR. A common partitioning used in current telephone systems is the concept of an area code. For example, if the user's number starts with 415, MSCs will know to send their queries to the HLR database in San Francisco. However, this approach has several problems that we outline in the next section.

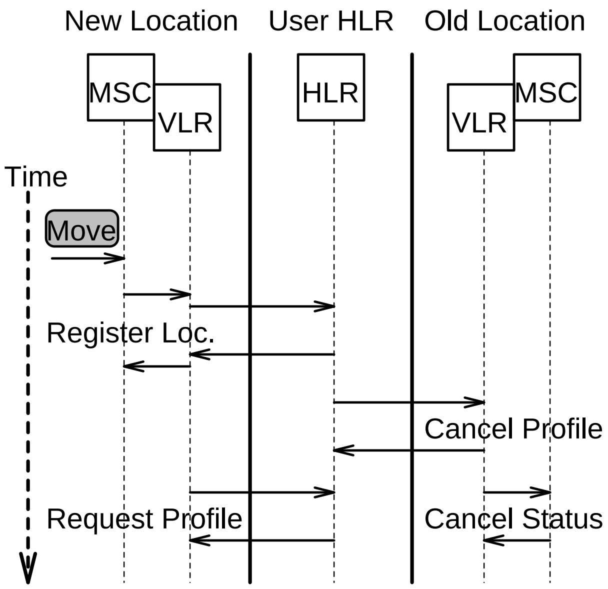

Figure 4: HLR/VLR Update Messages

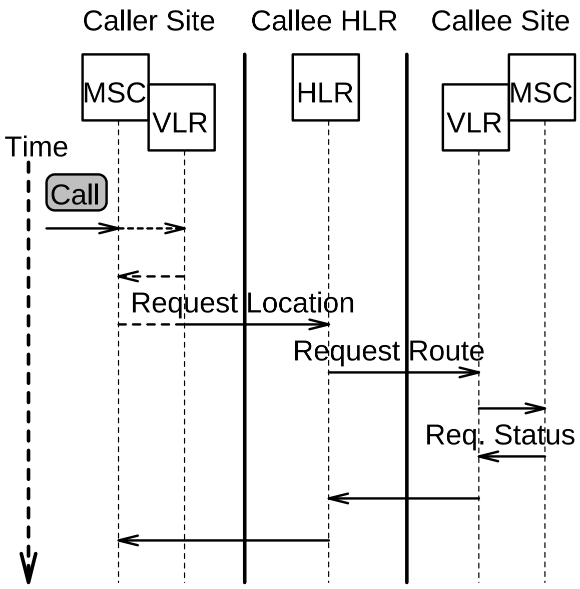

Figure 5: HLR/VLR Lookup Messages

3.2 Drawbacks

The IS-41 and GSM standards follow a simple approach to handle user location lookup and location updates, described in the previous section. While this approach is effective for a small-scale wireless communication infrastructure (such as a city), it has several problems that render it impractical for wide-area wireless infrastructures. We outline some particular problems with this approach that lead to higher response times than necessary, inflexible systems, and other undesirable anomalies.

Since profile lookups query the callee's HLR as a default, cross-country calls may be slow due to high network latencies. This problem can be reduced with selective replication and/or caching, however a more serious problem arises in the HLR/VLR scheme: If a caller makes a call to a callee whose HLR is distant from the current location of the caller, the MSC of the callee will query an HLR over a trans-continental network connection, even if the callee is physically located in the next registration area. In the case of remote network or HLR failures, physically adjacent users may not be able to communicate at all, even though the local network would be perfectly capable of handling their call.

Another problem with the HLR/VLR scheme is that user numbers are permanently bound to home locations due to the static partitioning of numbers to HLRs, as outlined in Section 3.1. Hence, if a user moves permanently from New York to San Francisco, the user is forced to change his number since his previous home location is across the country. This approach conflicts with one of the goals in wireless infrastructures—providing *location-independent* numbering so that users may keep the same number throughout their life. Hence, the HLR/VLR mechanism does not offer ideal support for user mobility in a wide-area network. In the next section we present alternative approaches, including our hierarchical scheme, which alleviate some or all of the problems outlined above.

4 Other Data Management Schemes

In this section, we first develop our new profile management scheme, which uses a hierarchy of autonomous databases to store user profiles. We then describe some "optimal" profile management schemes that are impractical to implement but are useful for base-line performance comparisons with our technique and others (see Section 7).

Hierarchical organization of data is not a new concept in distributed databases [OV91], nor is it unknown for profile management in wireless systems [AP91, Wan93, KVP94]. Our novelty arises from the following important factors, along with our extensive performance analyses reported in later sections.

1. We propose, for the first time, a simple hierarchical organization of databases that supports location independent numbering.

2. Our scheme is entirely compatible with the current IS-41 and GSM signaling standards. This compatibility is a very useful feature, since it is impractical to assume that every MSC will change to our data management technique immediately.

3. Our hierarchy scales, maintaining low database access and update rates at each layer within the hierarchy. In Section 7, we see that peak database access and update rates even for very large numbers of subscribers are well within current database capabilities.

4.1 Hierarchical Profile Management

The hierarchy contains databases at the leaves that store profiles of users located in a given registration area, as do VLR databases. Databases in the higher levels of the hierarchy store user numbers along with pointers to the lower level databases that store the user profile or in turn point to lower databases, and so on. We have a conceptual root database that stores a pointer for every wireless user. As in conventional databases, the root is distributed into several databases so that no one database needs to store all user profile pointers or service all root level queries and updates. Figures 6 and 7 show pseudocode for the lookup and update algorithms in our hierarchical scheme. Using our simple hierarchical scheme and these lookup and update operations, we achieve location independent numbering, system scalability, and significant improvements in some key performance measures (see Section 7). More precise pseudocode and other details for the hierarchical scheme can be found in [JLWC95].

```
Loc Lookup(calleeID) {
  loc = Query(calleeID)
  if (loc is null) {
    if (thisDB is rootLevelDB)
      return BAD
    else
      return parentDB->Lookup(calleeID)
  }
  else {
    childDB = ChildDB(loc)
    return childDB->Certify(calleeID)
  }
}

Loc Certify(userID) {
  loc = Query(userID)
  child = ChildDB(loc)
  if (child is not MSC)
    return child->Certify(userID)
  return loc
}
```

Figure 6: Hierarchical Profile Lookup

```
Update(userID, prevLevel) {
  loc = Query(userID)
  Update(userID, prevLevel)
  if (loc is null)
    parent->Update(userID, thisDB)
  else {
    child = ChildDB(loc)
    child->Canceldown(userID)
  }
}

Canceldown(userID) {
  loc = Query(userID)
  child = ChildDB(loc)
  if (child is not MSC)
    child->Canceldown(userID)
}
```

Figure 7: Hierarchical Profile Update

As an example, in Figure 8 we show how a profile lookup occurs when user B calls user A. The figure shows copies of user profiles stored at the leaves of the hierarchy, while higher level databases store pointers to all users located in the registration areas corresponding to the lower level databases. The query propagates, following the dark lines in the figure, through the hierarchy from the MSC of B until user A is found. In Figure 9 we show an example of user A moving to an adjacent registration zone. We see the database (at the lowest level of the hierarchy) in A's new

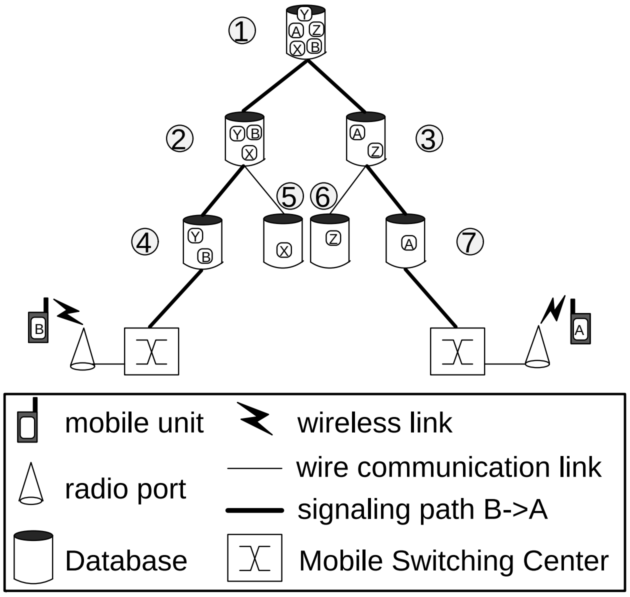

Figure 8: Lookup Signaling Path

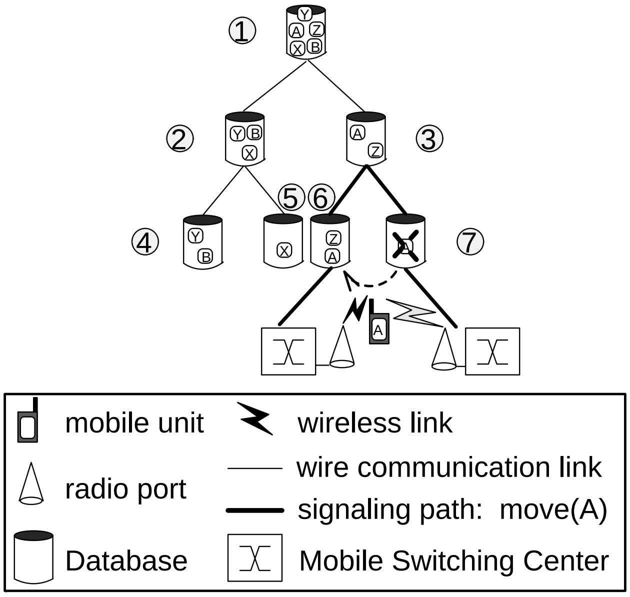

Figure 9: Update Signaling Path

location acquiring a copy of the user profile from A's old location. In this case, only the database at the next higher level needs to update its pointer for A to A's new location. In general, databases at even higher levels may need to be updated in order to point to the new location of a user when a user moves into a registration area not serviced by his previous database. (Essentially, we need to update all databases on the path to the least common ancestor.) In the worst case, e.g., if a user moves from the registration area served by DB 5 to the registration area served by DB 6, the pointers in databases 1, 2, and 3 need to be updated. In practice, we find that the number of databases to update is low due to "locality" in user mobility (see Sections 6 and 7).

In this paper, we concentrate on evaluating our technique with respect to the following performance measures: number of system-wide database queries and updates per second, number of individual database queries and updates per second, bandwidth required to communicate between databases, average number of link hops for inter-database messages (latency), and total database storage required. We have performed a comprehensive study of these performance aspects in our architecture (see Section 7), and we see that our technique performs very well due to locality in user calling and mobility patterns (see Section 6).

4.2 "Optimal" Profile Placement Policies

We now propose three other simple profile management techniques that are optimal for one or more of the performance measures discussed above. Since these techniques provide lower bounds for the relevant performance measures, it is useful to see how the current schemes and our new hierarchical scheme compare to these lower bounds.

1. **Centralized Database:** In this scheme, all user profiles are stored in a single database. Queries and updates are performed against the profile copy at the central database. This scheme has the minimum number of global queries and updates since a single database is used for storing all information. This scheme also requires minimum overall storage since there is exactly one copy of each user profile.

2. **Pure HLR:** In this scheme, user profiles are stored only at their permanent home location. Queries and updates are performed against the unique copy stored at the home of the user. This scheme is equivalent to the first scheme, with the addition of a static partitioning of the centralized database (based on, say, the area code field of user numbers) into several HLRs. Hence, this scheme has the same minimality characteristics as the centralized database.

3. **Full Replication:** In this scheme, user profiles are replicated in all databases throughout the infrastructure. Queries are performed against the local copy in the nearest database. When a user moves, updates are propagated to all databases. This scheme has the minimum number of global database queries, and the fastest possible location query response time.

Notice that the centralized database and full replication schemes also allow location independent numbering. In fact, our hierarchical scheme can be thought of as a "middle ground" between these two schemes, with potential for somewhat slower lookup, but not incurring the database bottleneck of the centralized scheme or the update overhead of full replication. Like the current standards, the pure HLR scheme is restricted to location dependent numbering.

We have not presented algorithms that minimize network traffic, since the minimum for this measure depends upon the interaction between user calling and mobility patterns, making it impossible to develop a single scheme that will be optimal under all conditions. Hence, we assume there is some abstract scheme that can produce the true minimum of zero network messages and zero network hops. This performance would occur, for example, if all queries were executed against a local database and updates did not need to be propagated to remote copies. Also notice that the three optimal algorithms we propose have significant overlap in terms of the lower bounds they achieve. We have implemented and simulated all of the above techniques, including IS-41, GSM, and our new hierarchical scheme, since they bring forward several trade-offs and new insights into building a nationwide wireless infrastructure (outlined in Section 7).

5 Pleiades Simulator

In this section, we describe the architecture of our extensible event-driven simulator, *Pleiades*, and we explain how indexing structures have been used to make our simulations faster and scale to millions of users. We then describe the many options built into Pleiades to simplify specification of complex database architectures, arbitrary network topologies, and geographical registration areas (called *sites*). We also outline the methods for specifying calling and mobility patterns for users.

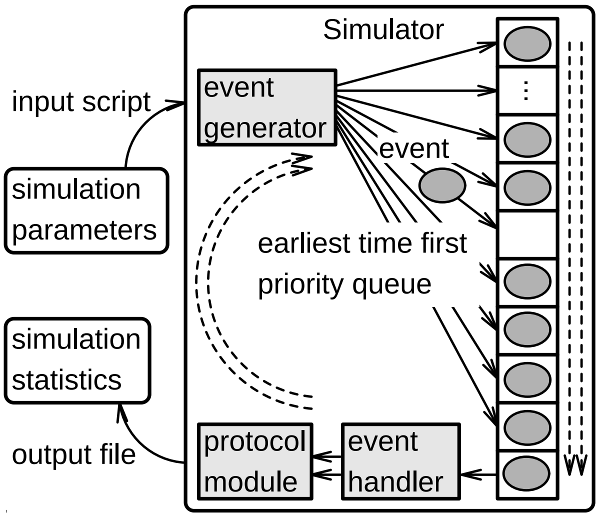

Figure 10: Pleiades Architecture

We developed Pleiades in C^{++} in just under 5000 lines of code. Simulation scripts specifying the calling patterns, mobility patterns, network topologies, and geographical entities range between 50 and 350 lines. We also have implemented all of the profile management techniques described earlier in approximately 100 lines of code each. Since Pleiades is object-based, we expect adding new techniques will require approximately the same amount of coding.

5.1 Architecture

Pleiades consists of an event generator, an event handler, an earliest-time-first priority queue, and a protocol module, as shown in Figure 10. Initially, the simulator constructs the geographical and network topologies according to a simulation script. The simulator then establishes a population of users, each of whom has a specified movement and calling pattern, from a set also defined in the simulation script.

The simulator cycles through a sequence of time windows in which call and move events are generated for the users according to the calling and movement patterns assigned to them in the simulation script. Each event is enqueued in proper temporal sequence. Once a full window of events is enqueued, the event processor passes them in sequence to the profile management module, which generates the simulation statistics sent to an output file. The statistics describe the movement and calling activities of the users, along with the database and network loads they incur.

We implemented the time priority queue as a specialized B$^+$-tree with access times logarithmic in the number of distinct time values stored, rather than in the number of events stored. Events generated with the same priority (time when they need to be dequeued) are stored in a linked list under the B$^+$-tree, and may all be extracted from the tree in a single operation. As a priority queue, the operations are `insert_with_priority_x`, and `get_top_priority`. These methods are implemented on the B$^+$-tree subclass as a straightforward insert with key value x, and a lookup followed by deletion of the first tree node and its linked list, respectively.

Our B^+-tree can service over 25,000 insertions and deletions per second on a 20 MIPS SPARC-station. This allows us to simulate very large user populations (currently just above 3 million users) within a given time period. These large-scale simulations enable analysis of more realistic performance characteristics of the different protocols than in previous simulation studies.

5.2 User Movement Model

In this section, we describe our user mobility model, which is powerful enough to handle a large class of user movement patterns. In Section 6 we describe one instance of this model we derived from eight months of vehicular traffic patterns in the San Francisco Bay Area for our simulation study in Section 7.

A simple Markovian model could have been used to generate user movements based on site border crossing probabilities, as in [BNK94]. Instead, we have developed a more detailed model, which includes the Markovian model as a special case. Most proposed profile management schemes optimize their performance for certain human behaviors. For example, movements *home* are important when studying schemes with home location registers (HLRs). Similarly, *roundtrip* movements are important when studying schemes with user profile replication. Our model generates user movements with varying distances and velocities and includes roundtrips and movements home.

Each user's movements are at all times defined by a *MoveType*. A MoveType consists of a number of *Move* objects. Each Move object models a particular class of movement behavior by defining its probability of occurrence, mean movement velocity and distribution, mean number of site crossings and distribution, and its move attribute. The move attribute indicates whether a Move is a *simple* move, *roundtrip*, *return home*, or *stationary*. Simple moves are those in which users roam around the geography according to site crossing probabilities. For moves with more than one site crossing, users do not return to the same site between adjacent moves. In other words, there are no simple movement loops of length two, since these moves are generated via roundtrip moves. moves home return users to their home locations via the most direct path.

Each MoveType has parameters describing a time period during which it is active, after which it is replaced by the next MoveType in the pattern. By changing a user's MoveType as a function of simulation time, we simulate temporal changes in user movement patterns, for example between morning rush hours and afternoon periods. The following sections illustrate how we parameterized our simulations with actual measured data.

5.3 User Calling Model

In this section, we describe our user calling model that can handle a large class of user calling patterns. In Section 6, we describe one instance of this model we derived from six months of calling traces from the Stanford Campus for our simulation study in Section 7.

Our call model generates call traffic for each individual user. We have adopted a model that takes into account different classes of calls, e.g., *short*, *long*, *retry*, along with ranking of people called most frequently (such as friends and associates). The model is thus divided into two parts: the *call traffic model* and the *callee distribution model*. We have implicitly assumed in our call model that callee distributions are not dependent on call traffic characteristics. We have verified this assumption by observing that low correlation exists between callers' average calling rates and their observed callee probabilities [LJCW95].

The call traffic model describes how often network users place calls to other users and the duration of each call. Very little is known about the traffic characteristics of future wireless networks. However, in fixed telephone networks, call traffic is modeled very accurately by a Poisson arrival process and exponentially distributed call durations. Our call traffic model is an extension of this basic model to handle mobility. It generates calls for different classes of traffic, and it models changes to user behavior through time.

The callee distribution model characterizes how callees are generated for each call. It is an important modeling issue because of its effects on system performance, particularly for schemes with caching or data replication, such as [BI92]. In the past, public telephone service providers were able to indirectly model callee distributions by measuring the cumulative call traffic into each service area. However, user mobility in wireless networks invalidates this approach. We model callee distributions by maintaining, for each caller, a list of the people they call most often with the probabilities of making calls to each of them. When a call is generated, the callee is selected either randomly from the set of all users, or from the user's callee list according to its probability distribution. Arbitrary callee distributions can be simulated by different initializations of the callee lists and their probability distributions.

6 Data Sources

6.1 Call Traffic Data

We have analyzed call traffic data [Tel95] for a six-month period from our university telephone exchange. This exchange serves the entire campus, including university offices, student housing, and residential households. The data was pre-processed at the source to protect the anonymity of the callers and callees. The data set covers 19,605 distinct callers and contains encrypted caller and callee identifications, time of call, and call duration for each outbound call. While this data constitutes a specialized sample of call traffic, the composition of our campus allows us to infer call traffic patterns for both business and residential settings.

Figure 11 shows average call volume over a 24-hour period. Each value for a given time is obtained by averaging over the corresponding time for each of the days in our data set. We observe that there are essentially three periods of call activity during a typical weekday. The first corresponds to night, 12–7 a.m., when there is very little activity. The second spans the regular business hours, 7 a.m.–6 p.m. The last constitutes the off-peak period during the evening hours.

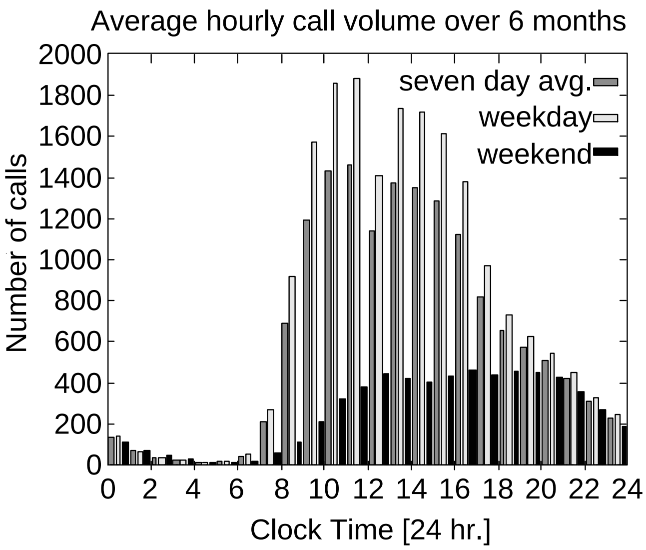

Figure 11: Average Outbound Calls Per Hour

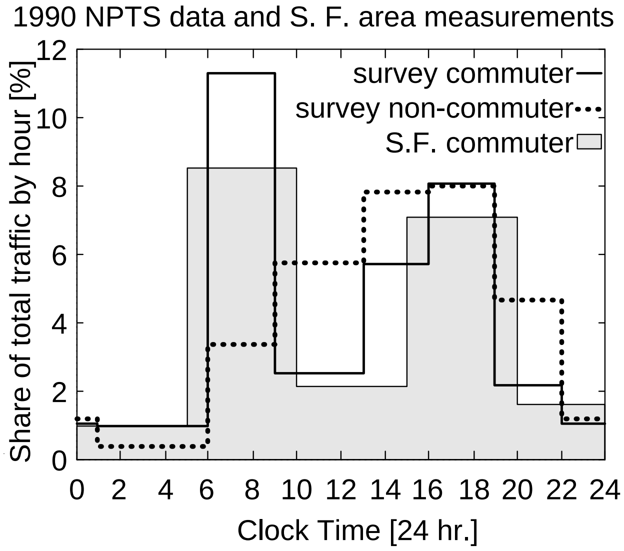

Figure 12: Estimated Movement Patterns

We have calculated empirical callee distributions by ranking callees according to the frequency with which they are called during reference time periods of one day, one week, and one month (four weeks). Callee rank is defined over the time period and is specific to a (caller, callee) pair. A rank k callee is the person a caller contacts k^{th} most frequently within the period. Figure 13 is a log-log plot of calling probability P_r versus callee rank. We observe that the mean probabilities obey a *power law* (or *generalized Zipf's law*) at all three reference time periods: $P_r \simeq \frac{A}{r^p}$ where A is the scaling parameter and p is the exponent parameter. We also find that 90% of all callers in our sample can be serviced with callee list lengths of 3, 9, and 25 for the three above time periods, respectively.

When initializing our callee lists, we also want to know the distributions around the average call probability, to accurately model the range of behaviors. For example, there should be users who make calls only to one person, i.e., $P_1 = 1.0$. In every data set we examined, the empirical distribution is well approximated by a normal distribution with the same mean and variance. Figure 14 shows the cumulative distributions and their fits to normal distributions. Relative call probabilities to the rank r callee follow a similar distribution for all reference time periods [LJCW95].

6.2 Vehicle Traffic Data

Our modeling of user movement behavior is based on a transportation survey, the *NPTS* [HY93, Kit95], conducted in 1990. This data is similar to vehicle traffic statistics for Europe [ECM95], in this case taken from roadside measurements. In addition, we have obtained actual movement statistics [MTC90] from the entire San Francisco Bay Area collected over an eight month period in 1989-1990 to correlate with survey data.

We begin with time-of-day traffic volume patterns using the above data. Figure 12 summarizes

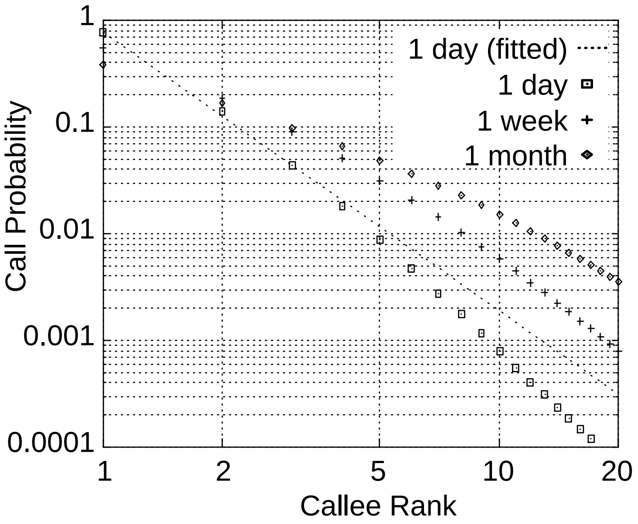

Figure 13: Call Probability vs. Callee Rank

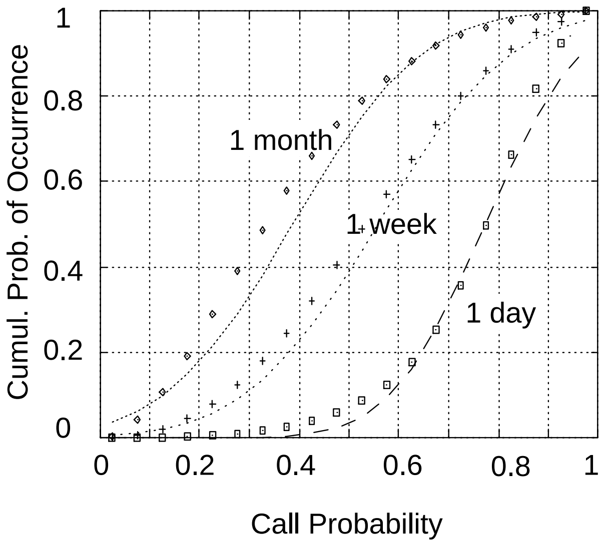

Figure 14: Prob. Distribution to 1st Callee

Trip Purpose	% of Trips	Avg. Trip Length		Avg. Velocity	
		miles	site ßXing	MPH	site/hr
to/from work	20.2	10.65	2	31.3	6.3
work-related	1.4	28.20	6	81.3	16.3
personal	52.9	6.74	1	28.3	5.7
social/etc.	25.3	11.53	2	39.2	7.8
vacation	0.2	218.22	(44)	261.5	(52.3)

Table 1: Movement Categories

the results. In the figure, the patterns *survey commuter* and *survey non-commuter* are derived from [HY93] and the measured pattern *S.F. commuter* from [MTC90]. Both measured and surveyed data suggest that commuter movement patterns can be modeled using five MoveTypes, one each for: (1) the late night low activity period between midnight and 6 a.m.; (2) the morning rush hour, 6–9 a.m.; (3) daytime roaming activities during business hours; (4) evening return commutes, 3–7 p.m.; (5) evening activities between 7 p.m. and midnight. The figure also indicates that non-commuters can be described in three MoveTypes: (1) a late night low activity period, 12–9 a.m.; (2) a period of gradually increasing activity, 9 a.m.–6 p.m.; (3) a moderate activity period from 6 p.m. to midnight.

We derived our estimates for move distances in Table 1 using NPTS survey data relating to mode of transportation, travel distance, and travel duration Distances in the simulation geography are represented at the granularity of the Site object, which for our simulations has an average radius of five miles.

The simulated commuter movements for the morning and evening rush hours use the *to/from* *work* figures. The *work-related* figures are part of the movement characteristics of commuters during the business hours. Data for *personal* and *social/etc.* movements are used to specify non-commuter movements, and commuter movements during non-business hours. The *vacation* category requires a different approach because mobile users turn off their equipment during the air travel typical of this category. We have not yet incorporated such movements into our study.

7 Simulation Results

7.1 San Francisco Bay Area Simulation

We performed our simulation on a geography that accurately models the San Francisco Bay Area. A map of the entire Bay Area is provided in Figure 15 for the reader's reference. The Bay Area consists of nine counties with a 1990 population of 6,023,877 [UpC94], and is serviced by four area codes. Regions corresponding to different area codes are represented by different shades in Figure 15; bridges, freeways, ferries and public transportation systems are also included for later reference. Figure 16 is an overlay map depicting the relationship between our simulation model and the physical geography [USG95]. Registration areas are represented as polygons in this figure. We simulated a 3-level hierarchy of databases. Small dots in the middle of polygons (registration areas) represent the databases servicing that area. Medium sized dots are databases servicing a set of lower level databases. The four large dots indicate the distributed root. Network links are represented by lines connecting the database dots.

In our simulation, we populated registration areas with users based on 1990 census information from [UpC94]. We divided our user population into 41% commuters and 59% non-commuters. (This proportion is derived from the national average in [HY93] and the peak-to-total traffic figure for the Bay Area in Figure 12.) We specified connectivities between transportation routes such as highways and bridges. Using traffic volume statistics from [MTC90], we estimated movement between area codes and fine-tuned our simulation parameters to produce similar large scale movement behavior.

We assumed that about 50% of the Bay Area residents will subscribe for wireless connectivity. We simulated these 3,025,000 residents for a typical 24-hour working day. We performed the actual simulations for a 48 hour period in order to eliminate all simulation transients, and we report performance results for the second 24-hour period. Each simulation generated 22.85 million move and 172.66 million call events, and averaged 11 hours to simulate 48 hours worth of traffic on an HP 9000/755/99 workstation with 512 MB RAM.

7.2 Performance of Profile Management Techniques

In this section, we first consider the system-wide database query loads, update loads, and network messaging loads for the different profile management techniques. We then study peak loads at databases in a sample set of registration areas. We conclude by summarizing the performance of the various techniques.

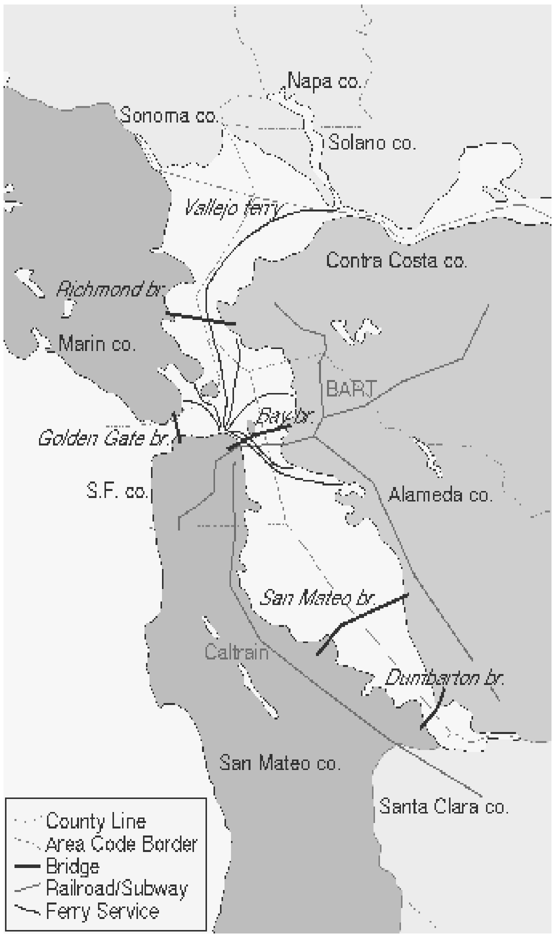

Figure 15: San Francisco Bay Area

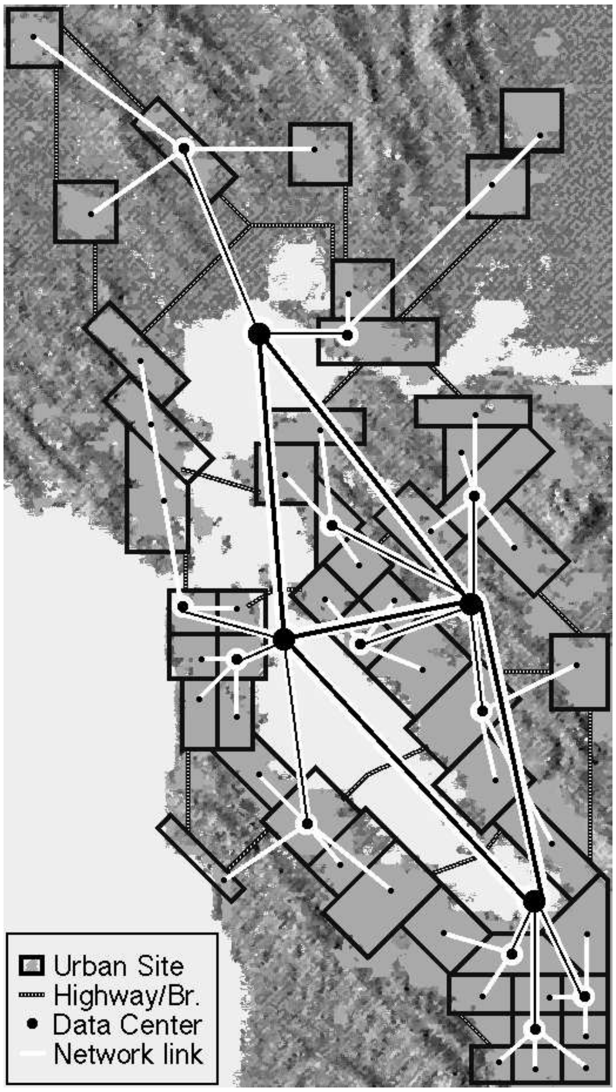

Figure 16: Simulation and Network Topologies

DB|net ops/s	Loc. Dependent		Location Independent Numbering		
	Pure HLR	HLR/VLR	Central DB	Full Replication	Hierarchical
DB Lookups	2304	4745	2304	2304	7762
DB Updates	284	741.2	284	25560	825.5
DB Totals	2529	5304	2529	27500	8379
Network Msgs	3045	4401	6360	26610	6107
Network Hops	8806	11000	15980	107900	6270

Table 2: Summary of Peak Protocol Requirements

7.2.1 Global Performance

Table 2 reports the peak levels of database operations and network signaling for each profile management scheme. Note that the peak for database lookups does not occur at the same time as the peak for database updates. Hence, the peak for total database load is not simply the sum of the peaks for lookups and updates.

Figures 17 and 18 contain log plots showing the evolution of database loads in the simulated 24-hour period for all five profile management techniques. Figures 19 and 20 show the total network signaling load. As expected, the centralized database, pure HLR, and full replication techniques have the same global database query characteristics and are lower bounds on the number of database queries. One of the more interesting results is that our hierarchical scheme incurs the least number of network hops. We have observed in informal cross-country experiments that latency due to database lookups is very low compared to network latency in multi-hop situations. Thus, we expect that our hierarchical scheme should have very good response times for call setup compared with the other schemes. The penalty is in increased global database loads, however we see in the next section that this increase should not pose a significant problem.

7.3 Local Performance

We now consider the performance of the various protocols with respect to activity at individual databases. Refer to Table 3. We present numbers for the Palo Alto registration area since it is close to the overall performance averages, and for Downtown San Francisco near the busy Bay Bridge since it has the peak database and network loads among all registration areas. We also present results for San Mateo and for the 510 area code root because they correspond to a second level and part of the root in our hierarchy, respectively. The 510 area code root is the most active among all the distributed roots because it services the largest population (about 1 million users).

We observe that individual site performance of the protocols follows the same trend as the global figures. In the hierarchical technique, root level databases are expected to have the highest database activity. From our results, we see that the 510 area code root database requirements (687^a

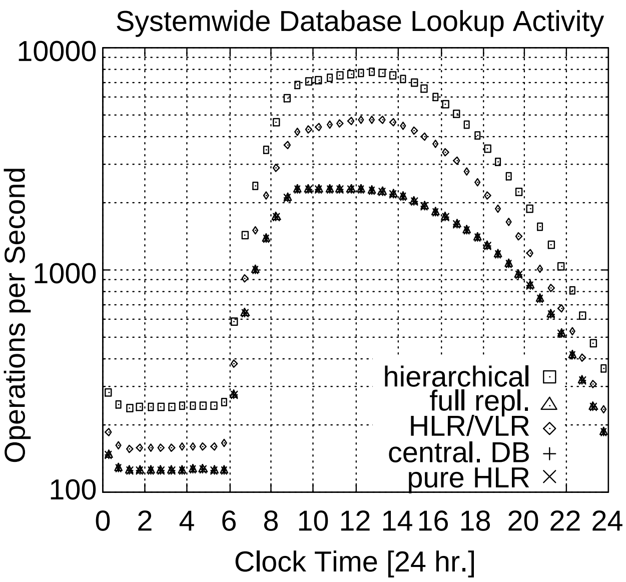

Figure 17: Profile Lookup Rate

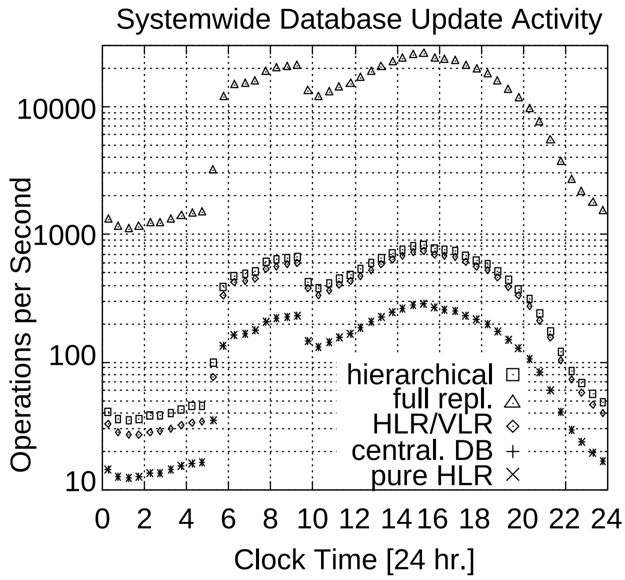

Figure 18: Profile Update Rate

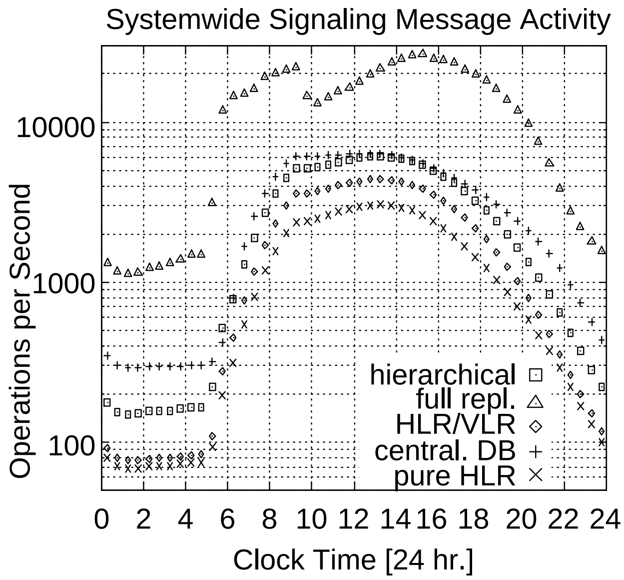

Figure 19: Network Signaling Messages

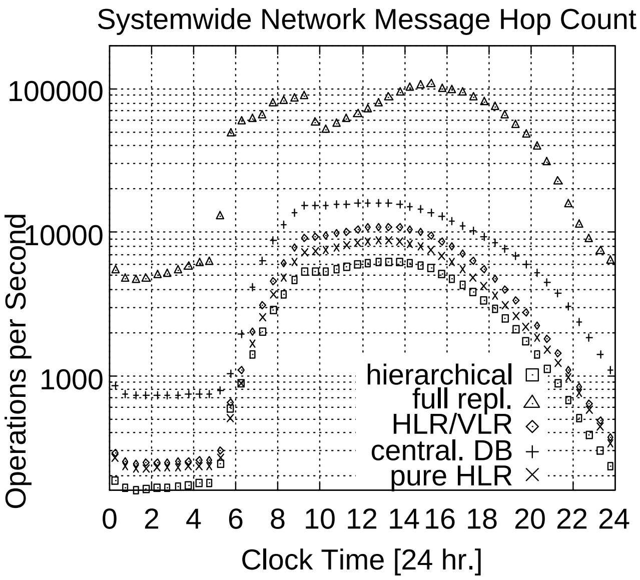

Figure 20: Message Hop Count

	ops/s	Pure HLR	HLR/VLR	Central DB	Full Repl.	Hierarchical
Palo Alto	DB Lookup	38.2	48.2	n.a.	46.3	80
	DB Update	4.30	14.2	n.a.	284	6.16
	Network Msgs	62.5	89.9	84.7	323	43.7
	Network Hops	178	224	250	2254	**45.4**[c]
Downtown	DB Lookup	76.7	86.5	n.a.	83.6	141
	DB Update	9.06	29.2	n.a.	284	12.0
	Network Msgs	111	160	150	1120	77.2
	Network Hops	323	401	446	4609	**81.2**[d]
San Mateo	DB Lookup	38.5	37.4	n.a.	36.6	212
	DB Update	4.52	10.0	n.a.	284	28.0
	Network Msgs	44.0	64.4	60.2	354	187
	Network Hops	90.9	111	124	1134	**187**[e]
(510) Root	DB Lookup	n.a.	n.a.	2304	n.a.	**687**[a]
	DB Update	n.a.	n.a.	284	n.a.	**65.0**[b]
	Network Msgs	n.a.	n.a.	2304	n.a.	735
	Network Hops	n.a.	n.a.	5460	n.a.	**735**[f]

Table 3: Peak Site Requirements Per Second

Hierarchical vs.	Lookup	Update	Messages	Hops	Low Store	High Store
HLR/VLR	1.64	1.11	1.39	0.57	1.2	1.13
Optimal	3.37	2.91	(2.01)	(0.71)	2.4	2.13

Table 4: Relative Peak Requirements Summary

lookups + 65^b updates per second), the highest among all distributed roots, fall well within the capability of current databases [GR93, TPC95]. We also see that the average number of network hops for messages is again lowest or close to lowest (45.4^c, 81.2^d, 187^e, and 735^f) for our hierarchical scheme.

7.4 Summary of Simulation Results

We first summarize how our hierarchical scheme compares with the other profile management schemes. We then derive some more performance numbers to indicate why we believe that our hierarchical scheme is practical.

Table 4 summarizes Table 2 and shows how our scheme performs relative to HLR/VLR and against the lower bounds established by the "optimal" algorithms described in Section 4.2. The numbers in the table indicate a multiplier for the hierarchical scheme as compared with the other schemes for each performance measure. In the case where lower bounds were not available, we have compared our scheme with the best of the optimal schemes and indicate such numbers within parentheses. In addition to the earlier reported performance measures, we also report the relative disk space required for the schemes. *Low Store* is the storage requirement assuming a conservative 100-byte profile size [PMG95] and 20 bytes for (ID, location) pairs; *High Store* is the storage requirement assuming 300-byte profile sizes as in the GSM standard.

We see from Table 4 that our hierarchical scheme suffers in terms of the number of database lookups and updates. But, as we saw in Table 3, the peak database loads at all levels in the hierarchy are well within current database capabilities. Our scheme also suffers from a higher network bandwidth requirement. However, rough calculations (based on the number of messages and message sizes) indicate that the system-wide bandwidth required for our hierarchical scheme is only about 11.18 Mbits/sec for 100-byte profiles and 20.50 Mbits/sec for 300-byte profiles. In today's world of gigabit networks, again we believe our scheme's bandwidth requirement is not a problem.

On the other hand, notice that the hierarchical scheme gives the lowest average number of hops per message, indicating (as explained earlier) that it is likely to be very fast for call setup. In addition, the hierarchical scheme scales well for all the above measures, whereas the "optimal" schemes all conceal some form of scalability limitations. In summary, we believe that our new hierarchical scheme is very promising, since it can support attractive features for users such as location independent numbering, as well as providing fast call setups with relatively minor performance penalties.

8 Conclusion

We have developed a new hierarchical technique for profile management in wireless networks. The new technique is scalable, compatible with current data management standards, and provides attractive user features such as location independent numbering and fast call setup. We have evaluated our technique against current data management standards and other schemes. To do so, we have developed the Pleiades simulator, which allows us to accurately model geography, network structure, and user activity for large user populations. Our simulation model was derived from real calling and traffic data, and our simulations were performed on a model of a real geographical area (the San Francisco Bay Area). Our experimental results indicate that while our hierarchical scheme does incur some performance penalties over other (less flexible) schemes, the performance requirements still lie well within current database and network technologies.

Acknowledgments

Thanks to Bora Akyol and Wilburt Labio for useful comments, Susan Phillips and the Communications Services at Stanford for call traffic data, and Nesrin Basoz for vehicle traffic data.

References

[AP91] B. Awerbuch and D. Peleg. Concurrent online tracking of mobile users. *Computer Communication Review*, 21(4):221–233, September 1991.

[BI92] B. R. Badrinath and T. Imielinski. Replication and mobility. In *Second Workshop on the Management of Replicated Data*, pages 9–12. IEEE, November 1992.

[BNK94] A. Bar-Noy and I. Kessler. Mobile users: To update or not to update? In *INFOCOM '94*, pages 570–576. IEEE, June 1994.

[ECM95] ECMT. *European Transport Trends and Infrastructural Needs.* OECD publications, 2, Rue Andre-Pascal, 75775 Paris CEDEX 16, France, 1995.

[EIA91] EIA/TIA IS-41.3 (Revision B). *Cellular Radiotelecommunications Intersystem Operations*, July 1991.

[GR93] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kauffman, 1993.

[HJM94] 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.

[HP90] J. L. Hennessy and D. A. Patterson. *Computer Architecture A Quantitative Approach.* Morgan Kauffman, 1990.

[HY93] P. S. Hu and J. Young. *1990 NPTS Databook: Nationwide Personal Transportation Survey.* Federal Highway Administration, November 1993.

[IB92] T. Imielinski and B. R. Badrinath. Querying in highly mobile distributed environments. In *VLDB '92*, pages 41–52, 1992.

[IB93] T. Imielinski and B. R. Badrinath. Data management for mobile computing. *SIGMOD Record*, 22(1):34–39, March 1993.

[Jag94] H. V. Jagadish. Databases for networks. In *ACM SIGMOD*, page 522, June 1994.

[JLLM94] 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.

[JLWC95] J. Jannink, D. Lam, J. Widom, and D. C. Cox. Data management for user profiles in wireless communications systems. Technical report, Stanford University, Computer Science Dept., October 1995. ftp://www-db.stanford.edu/pub/jannink/1995/profile/.

[Kit95] R. Kitamura. *Time-of-Day Characteristics of Travel: An Analysis of 1990 NPTS Data*, chapter 4. Federal Highway Administration, February 1995.

[KVP94] P. Krishna, N. H. Vaidya, and D. K. Pradhan. Location management in distributed mobile environments. In *Third International Conference on Parallel and Distributed Information Systems*, pages 81–88, September 1994.

[LJCW95] D. Lam, J. Jannink, D. C. Cox, and J. Widom. Modeling location management for Personal Communication Services. Technical report, Stanford University, Computer Science Dept., October 1995. ftp://www-db.stanford.edu/pub/jannink/1995/model/.

[MP92] M. Mouly and M.-B. Pautet. *The GSM System for Mobile Communications.* Palaiseau, France, 1992.

[MTC90] 1990 commute summary. Metropolitan Transportation Commission, Lotus 123 format spreadsheet, August 1990. S.F. Bay area transportation measurements.

[OV91] M. T. Oszu and P. Valduriez. *Principles of Distributed Database Systems.* Prentice Hall, 1991.

[PMG95] G. P. Pollini, S. Meier-Hellstern, and D. J. Goodman. Signaling traffic volume generated by mobile and personal communications. *IEEE Communication Magazine*, pages 60–65, June 1995.

[SW95] N. Shivakumar and J. Widom. User profile replication for faster lookup in mobile environments. In *1st ACM International Conference on Mobile Computing and Networking*, 1995.

[Tel95] Telephone call traffic data set, 3/95–9/95. S. Phillips, personal communication, October 1995. Encrypted ID numbers.

[TPC95] Transaction processing benchmarks. http://www.ideas.com.au/bench/bench.htm, 1995. Monthly TPC-A, TPC-B & TPC-C benchmark results.

[UpC94] 1990 U.S. census demographic summaries. UpClose Publishing, http://www.upclose.com/upclose/, 1994. Population of nine S.F. Bay area counties.

[USG95] 1990 urban land use in central California. U.S. Geological Survey, http://geo.arc.nasa.gov/usgs/images/urban2.gif, 1995. Urbanization in S.F. Bay area.

[Wan93] 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.

[Wan94] D. C. C. Wang. A survey of number mobility techniques for PCS. In *International Conference on Universal Personal Communications, ICUPC '94*, pages 340–344. IEEE, 1994.

[WK95] M. Wang and W. J. Kettinger. Projecting the growth of cellular communications. *Communications of the ACM*, 38(10):119–122, October 1995.

A Notes, Definitions and Pseudocode

A.1 Preliminary notes

Basic definitions:

- `xyz` is a variable
- `Abc()` is a function call
- `=` is the assignment operator
- `->` is the usual message passing method operator

MSC algorithms:

- `AuthKey Register(userID, thisMSC)` /* user registration */
- `Void Cancel(userAuthKey)` /* user deregistration */
- `Stream Connect(calleeAuthKey)` /* create call circuit */
- `Void HUP(userAuthKey)` /* tear down call circuit */
- `Stream Call(userAuthKey, calleeID)` /* initiate call setup */
- `AuthKey Move(userAuthKey, thisMSC)` /* move re-registration */

MSC definitions & assumptions:

- IDs have one DB address embedded in them
- AuthKeys have one or more DB addresses and/or MSC address embedded in them
- DBs do the work of making IDs and AuthKeys and embedding addresses in them
- `RegisterDB()`,
- `UserVerifyDB()`,
- `CallSetupDB()`,
- `ConnectMSC()`,
- `BillingDB()` are primitives that allow the MSC to extract a network address

DB algorithms:

- `AuthKey Register(userID, thisMSC)` /* similar to `Update()` */
- `Void Deregister(userAuthKey)` /* similar to `Update()` */
- `ACK StartBillingTrans(userAuthKey)` /* separate from rest of signaling */
- `ACK EndBillingTrans(userAuthKey)` /* separate from rest of signaling */
- `AuthKey Authenticate(userAuthKey)`
- `AuthKey Lookup(calleeID)`
- `AuthKey Update(userAuthKey, thisMSC)`
- `Cancel(userAuthKey)` /* cancels old authorization value */
- `Canceldown(currAuthKey)` /* recursive authorization cancellation to MSC */
- `Certify(currAuthKey)` /* profile certification to callee MSC */

DB definitions & assumptions:

- If a DB is in a hierarchy it knows its parent DB
- If a DB does full replication it knows its peer DBs
- `ExtractID(userAuthKey)` extracts the embedded user ID in the AuthKey
- `QueryAuthKey(userID)` queries the DB looking for local authorization
- `UpdateAuthKey(userID, newAuthKey)` stores a new authorization value
- `MakeAuthKey(userID, prevLevel)` creates a local authorization value
- `HomeDB(calleeID)`,
- `HomeRootDB(calleeID)`,
- `OldDB(userAuthKey)`,
- `ChildDB(calleeAuthKey)` extract a database address stored in the AuthKey

A.2 MSC Pseudocode

Pseudocode presented below is valid for all of the five database organizations presented in the paper.

```
AuthKey Register(userID, thisMSC)
{
  regDB = RegisterDB(userID)
  if (regDB is BAD)
    return BADuser

  userAuthKey = regDB->Register(userID, thisMSC)
  if (userAuthKey is BAD)
    return BADuser
  else
    return userAuthKey
}
```

```
Void Cancel(userAuthKey)
{
  deregDB = RegisterDB(userAuthKey)
  deregDB->Deregister(userAuthKey)
}
```

```
Stream Connect(calleeAuthKey)
{
  hookup = PageMobile(calleeAuthKey)
  if (hookup is INACTIVE)
    return BAD
  else if (hookup is BUSY)
    return BUSY
  else {
    billDB = BillingDB(calleeAuthKey)
    billDB->StartBillingTrans(calleeAuthKey)
    return hookup
  }
}
```

```
Void HUP(userAuthKey)
{
  billDB = BillingDB(userAuthKey)
  billDB->EndBillingTrans(userAuthKey)
}
```

```
AuthKey Move(userAuthKey, thisMSC)
{
  moveDB = RegisterDB(userAuthKey)
  userAuthKey = moveDB->Update(userAuthKey, thisMSC)
  if (userAuthKey is BAD)
    return BADuser
  else
    return userAuthKey
}
```

```
Stream Call(userAuthKey, calleeID)
{
  userDB = UserVerifyDB(userAuthKey)
  if (userDB is BAD)
    return BADuser
  if (userDB->Authenticate(userAuthKey) is BAD)
    return BADuser

  setupDB = CallSetupDB(calleeID)
  if (setupDB is BAD)
    return BADcallee
  calleeAuthKey = setupDB->Lookup(calleeID)
  if (calleeAuthKey is BAD)
    return BADcallee

  connectMSC = ConnectMSC(calleeAuthKey)
  hookup = connectMSC->Connect(calleeAuthKey)
  if (hookup is BAD)
    return incompleteCallMessage
  else if (hookup is BUSY)
    return busySignal
  else {
    billDB = BillingDB(userAuthKey)
    billDB->StartBillingTrans(userAuthKey)
    return hookup
  }
}
```

A.3 Additional DB Pseudocode

```
AuthKey Authenticate(userAuthKey)
{
  userID = ExtractID(userAuthKey)
  currAuthKey = QueryAuthKey(userID)
  if (calleeAuthKey is EMPTY)
    return BAD
  else
    return currAuthKey
}
```

A.3.1 'Optimal' Schemes' Lookup & Update

```
AuthKey Lookup(calleeID)
{
  calleeAuthKey = QueryAuthKey(calleeID)
  if (calleeAuthKey is EMPTY)
    return BAD
  else
    return calleeAuthKey
}
```

```
AuthKey Update(userAuthKey, thisMSC)
{
  userID = ExtractID(userAuthKey)
  newAuthKey = MakeAuthKey(userID, thisMSC)
  UpdateAuthKey(userID, newAuthKey)
    /* next 2 lines for Full Replication only */
    foreach peerDB
      peerDB->UpdateAuthKey(userID, newAuthKey)
    /* * * * * * * * * * * * * * * * * * * * */
  return newAuthKey
}
```

A.3.2 HLR/VLR Lookup & Update

```
AuthKey Lookup(calleeID)
{
  calleeAuthKey = QueryAuthKey(calleeID)
  home = HomeDB(calleeID)
  if (calleeAuthKey is BAD) {
    if (thisDB is home)
      return BAD
    else
      return home->Lookup(calleeID)
  }
  else {
    if (thisDB is home) {
      childDB = ChildDB(calleeAuthKey)
      return childDB->Certify(calleeID)
    }
    else
      return calleeAuthKey
  }
}
```

```
AuthKey Update(userAuthKey, thisMSC)
{
  userID = ExtractID(userAuthKey)
  home = HomeDB(userID)
  if (thisDB is home)
    newAuthKey = MakeAuthKey(userID, thisMSC)
  else {
    newAuthKey = home->Update(userAuthKey, thisMSC)
    oldDB = OldDB(userAuthKey)
    oldDB->Cancel(userAuthKey)
  }
  UpdateAuthKey(userID, newAuthKey)
}

AuthKey Certify(userAuthKey)
{
  userID = ExtractID(userAuthKey)
  return QueryAuthKey(userID)
}
```

A.3.3 Hierarchical Lookup & Update

```
AuthKey Lookup(calleeID)
{
  calleeAuthKey = QueryAuthKey(calleeID)
  if (calleeAuthKey is EMPTY) {
    if (thisDB is rootLevelDB) {
      home = HomeRootDB(calleeID)
      if (thisDB is home)
        return BAD
      else
        return home->Lookup(calleeID)
    }
    else
      return parentDB->Lookup(calleeID)
  }
  else {
    childDB = ChildDB(calleeAuthKey)
    return childDB->Certify(calleeAuthKey)
  }
}

AuthKey Certify(userAuthKey)
{
  userID = ExtractID(userAuthKey)
  currAuthKey = QueryAuthKey(userID)
  child = ChildDB(currAuthKey)
  if (child is MSC)
    return currAuthKey
  else
    return child->Certify(currAuthKey)
}
```

```
AuthKey Update(userAuthKey, prevLevel)
{
  userID = ExtractID(userAuthKey)
  currAuthKey = QueryAuthKey(userID)
  newAuthKey = MakeAuthKey(userID, prevLevel)
  UpdateAuthKey(userID, newAuthKey)
  if (currAuthKey is EMPTY) {
    if (thisDB is RootLevelDB) {
      home = HomeRootDB(userID)
      home->Update(newAuthKey, thisDB)
    }
    else
      parent->Update(newAuthKey, thisDB)
  }
  else {
    child = ChildDB(userAuthKey)
    child->Canceldown(currAuthKey)
  }
  return newAuthKey
}

ACK Canceldown(userAuthKey)
{
  userID = ExtractID(userAuthKey)
  currAuthKey = QueryAuthKey(userID)
  child = ChildDB(currAuthKey)
  if (child is MSC)
    return OK
  else
    return child->Canceldown(currAuthKey)
}
```