Optimization of a Signaling Hierarchy for Mobility Tracking in Personal Communications Networks

V. Anantharam M. L. Honig U. Madhow V. K. Wei

Abstract

Personal communications implies the ability to address a call to a *person*, rather than to a telephone or computer at a fixed location. Each user is assigned a unique name or personal number, and is able to access the wireline network (such as the public switched telephone network), independent of his or her location, by means of a wireless link. Call set-up therefore requires that the wireline network determine the current location of the user being called before the call is routed. Tracking mobile users requires *updates* at databases within the signaling network, and call set-up requires *accesses* at these databases. In order to distribute the load evenly among the databases, we propose a signaling scheme based on a hierarchy of databases. We consider the problem of how to configure this hierarchy to minimize the total number of database updates and accesses, given estimates of mobility and calling rates, and subject to constraints on the maximum access and update rate for each database. A dynamic programming solution to this problem is presented. We show that dynamic programming can also be used to minimize other cost functions such as the average call set-up time due to queueing delays in the network.

1 Introduction

A Personal Communications Network (PCN) relies on wireless links to a wireline network to support continuous voice and data communications among mobile users (or portable devices, such as computers). Each mobile user communicates with a stationary radio port, or *base station*, using a wireless link. Provided that the current location of a user is known, or can be found, by the wireline network, calls to a mobile user can be routed to the appropriate base station using the existing wireline network. The base station then transmits the call to the mobile user using a wireless link. [2]

[1]V. Anantharam is with the School of Electrical Engineering, Phillips Hall, Cornell University, Ithaca, NY 14853. M. L. Honig, U. Madhow, and V. K. Wei are with Bellcore, 445 South Street, Morristown, NJ 07960.

[2]"Personal Communications" also implies the ability of users to attach portable devices to different points of the wireline network. Wireless links are not necessarily used in such situations. However, the wireline network must maintain location information.

User mobility causes two main problems with call set-up and routing that are not encountered in a network containing only stationary users. Firstly, call set-up requires at least one *database access* to find the current location of the user being called. Secondly, if the network maintains a list of locations of each mobile user, then every location change requires one or more *database updates*. In addition, the communication links of the signaling network (which may be different from the information bearing network) must be able to accomodate the additional traffic generated by these updates and accesses. One possible solution is to use a single database which contains the locations of all users in the network. This database must then be updated (accessed) every time a user changes location (makes a call). This solution becomes infeasible if a single database cannot handle all the updates and accesses generated in the entire network, or if signaling traffic to and from this database becomes excessive.

In this paper we propose a hierarchical arrangement of databases for mobility tracking that reduces the access and update load on each database, relative to the centralized scheme just described. This scheme is motivated by recent studies which estimate the number of updates, queries, and volume of signaling traffic that will be generated in a PCN [6], [7]. In [6], it is assumed, as in GSM [5], that each user is identified with a particular node in the signaling network, called a Home Location Register (HLR), which contains the user's current location. Each location change (call set-up) then requires that the HLR be updated (accessed). This system becomes inefficient if a user travels far from his HLR, or if there is a relatively large amount of traffic between two HLR's far apart in the network. In these cases, update and query traffic may traverse several links in the signaling network, causing congestion. The hierarchical scheme presented here has the capability of greatly reducing the number of links over which signaling traffic must travel in these situations.

The mobility tracking scheme considered here assumes that the signaling network is a tree, and that users are located at the nodes of this tree. In some instances, the tree can be designed so that only its leaves correspond to nodes in the information bearing wireline network. In such a case users are located only at the leaves of the tree. In other instances, the signaling tree may overlap with the information network, and users can be located at any node in the tree. Each node in the tree routes signaling traffic, in addition to possibly containing a database which maintains a list of all users currently located at nodes in its associated subtree. The entry corresponding to a user in each database consists of a pointer to the appropriate database further down in the tree. A pointer to the node at which the user is located is contained in the lowest database above this node. Call set-up is therefore established by following a series of pointers to the current location of the user. When the user moves to a nearby location, only databases which are relatively low in the hierarchy need to be updated. Similarly, call set-up between nearby users need not access databases relatively high up in the hierarchy. The tree structure therefore serves to balance accesses and updates among databases. However, an associated disadvantage is that a single location change or call may cause updates or accesses at several databases.

The preceding tradeoff is captured by the following optimization problem. Given estimates of the rates of calls and movements between different nodes of the network, and given

an upper bound on the rate at which each database can be updated or accessed, find a placement of databases at nodes in the tree so as to minimize the *total* number of updates and accesses per unit time in the network. If the database capacity is sufficiently large, then the solution to this problem is the centralized scheme which places a single database at the root node. The cost criterion can be easily modified to reflect other performance measures, such as the delay in call set-up due to congestion at the communication links and databases. An optimization criterion which accounts for signaling delays can lead to a more complicated solution than the centralized scheme even when the databases have arbitrarily high update and access rates.

The type of signaling network considered here may cover geographical regions of various sizes, and may reflect geographical hierarchies. For example, the root node may cover an entire country, nodes at the next level in the tree may be gateways to networks covering states, nodes at the next level may be gateways to Metropolitan Area Networks, and so forth. Because cells in a PCN can be quite small, the wireline signaling network typically does not keep track of user locations down to the cell level. Rather, the nodes of the signaling network at which users can reside might represent wire centers or Local Area Networks (LANs). Once a user is identified to be at a particular node in the signaling tree, the particular cell in which he or she currently resides can be found, for example, by wireless broadcast.

The mobility tracking scheme presented here has been independently proposed in [10], although no attempt at optimization is described there. Hierarchical tracking and routing in mobile telephony has also been considered in [1] and [9], although the approaches taken in these references are different from that described in this paper. In [1], a distributed hierarchy for mobility tracking based on a graph-theoretic structure called "regional matching" is presented. That algorithm is based on distances between nodes; important parameters such as estimates of call and mobility rates are not used. In [9], a hybrid scheme which combines aspects of a signaling hierarchy with the notion of HLR is proposed, and a queueing analysis is presented for a model similar to the example presented in Section 4 of this paper. A preassigned database configuration is assumed, without any attempt at optimization.

Bar-Noy and Kessler [2] have recently considered the problem of selecting a subset of nodes in a tree, which represents the signaling network, so as to optimize the use of *wireless* resources for tracking mobile users. In contrast, our focus is on optimization within the *wireline* network. Despite this difference in application, the associated mathematical optimization problems turn out to be similar. Both can be solved using dynamic programming, and the complexity analyses for the algorithms are very similar.

A mathematical description of the mobility tracking model and the optimization problem considered are presented in Section 2. A dynamic programming solution to this optimization problem is presented in Section 3. Section 4 contains a simple example which illustrates the algorithm. In Section 5, we show that this dynamic programming algorithm can be used to optimize the placement of databases for a more general class of cost functions than the one originally considered. An example is given in which the cost function is the average delay in call set-up due to queueing at the databases and communication links. Finally, Section 6 contains our conclusions.

2 System Model and Problem Formulation

Given a signaling network represented by a connected graph, we construct a rooted spanning tree T of this graph, shown in Figure 1. The construction of T may be dictated by geographical considerations as well as by call and mobility patterns within the network, and is not considered here. For the remainder of this section, we assume that T is given. Each user has a unique identifier, or *name* (e.g., a personal phone number). Each node is capable of relaying messages up and down the tree, and some of the nodes also have databases. We assume that the root of T always contains a database. For simplicity of exposition, we also assume that users are located only at leaf nodes. The mobility tracking scheme presented here can be easily modified to accomodate users at internal nodes as well. A description of our system model follows.

Database Contents: Assume that a set of nodes has been selected as database sites. The database at a given node a contains one entry for each user located at any node in the subtree rooted at a. User x's entry states that either (1) "user x is at database b" where b is the next database node on the path from a to the node where x currently resides, or (2) "user x is at node b" if x is at node b, and there is no database on the path from a to b (not counting a).

We assume each leaf node has a list of local users. These lists are not considered databases in our terminology.

Call Set-up Functions of a Node with a Database: If node a has a database, and receives a message "call for user x", it attempts to find an entry for x in its database. If an entry "user x is at database (or node) b" is found, then a routes the message to b (say, by attaching the address of b to the message). If no entry for user x is found, then node a relays the request up the tree.

Call Set-up Functions of a Node without a Database: It simply relays the message up or down the tree, depending on whether the message comes from a descendant or ancestral node, respectively.

A call set-up message is successful once it reaches the leaf node which contains the called party. Consider a call set-up request originating from leaf node i for a user which resides at leaf node j. It first heads up the tree until it reaches a certain database node. From there the request heads down the tree to j. The turnaround node is the lowest database node which is an ancestor of both i and j. The message may also encounter other databases along the way, as specified above.

Example: Consider the signaling network shown in Figure 1. For illustration, assume that there are only four databases located at nodes a, b, c, and c_2, and assume that there are only six users: u, v, w, x, y, z at nodes 1, 2, 6, 3, 11, 12, respectively. The contents of databases are shown below:

Database a	Database b	Database c	Database c_2
u@Database b	u@Database c	u@*node* 1	$[EMPTY]$
v@Database b	v@Database c	v@*node* 2	
w@Database b	w@Database c	w@*node* 6	
x@Database b	x@*node* 3		
y@*node* 11			
z@*node* 12			

Example (Operations required for a call set-up): Suppose there is a call request for user y (who is at node 11) originating at node 9. The request is relayed up the tree by node e_2 to node c_2. Since database c_2 does not contain y, it relays the request up to node b_1, which in turn relays it up to node a. The database a contains the entry "y is at 11", so it routes the message down to node 11, the final destination, via the nodes b_1, c_3, and e_3. Thus, the turnaround node for the call set-up message is node a, the lowest database node which is an ancestor of both the originating node 9 and the destination node 11.

The task of mobility tracking is similar. Assume user x moves from node j to node k. Node j removes x from its list of local users, node k adds x to its list of local users, and node k initiates a message "user x moves to node k" and sends it up the tree. The message first heads up the tree until it reaches a certain database node, from there it heads down the tree to node j. This turnaround node is the same turnaround node as for a call set-up request from node k for a user at node j. The message is altered along the way, as specified below. The update sequence is confirmed as successful once the update message reaches node j.

Mobility Tracking Functions of a Node with a Database: There are two cases: (1) When database b receives a message "x moves to ℓ", it searches for user x. If not found, node b adds the entry "x is at ℓ" to its database and sends the message "x moves to b" up the tree. If an entry "x is at a" is found, node b replaces it with "x is at ℓ" and sends the message "x departs a" down the tree to node a. (2) When database b receives a message "x departs b", it must contain an entry for user x, say "x is at c". It discards this entry and sends a message "x departs c" to node c.

Mobility Tracking Functions of a Node without a Database: It simply relays the message up or down the tree, just as in call set-up.

Example (Operations required for a movement): Suppose user x moves from node 3 to node 9. Node 9 sends the message "x moves to 9" up the tree. The message is relayed up the tree by node e_2 to node c_2. Since database c_2 does not contain x, it adds the entry "x is at 9", and sends the message "x moves to c_2" up the tree. The message is relayed by node b_1 to node a. Database a replaces the entry "x is at b" by the entry "x is at c_2", and sends the message "x departs b" down the tree to node b. Database b deletes the entry "x is at 3", and sends a message "x departs 3" down the tree to node 3, via the nodes c_1 and d_1. This

serves as confirmation to node 3 that the appropriate modifications have been made to all the relevant databases in the tree.

An important issue is whether or not the preceding protocol can ever result in a deadlock or inconsistency. We do not examine this issue in detail; however, we briefly describe what happens if a call arrives in the middle of an address update. Assume that while user x moves from node 3 to node 6, a call is placed to user x. In the worst-case scenario, the call set-up request is routed via database b to node 3 before the address update message from node 6 reaches b. In this case, user x will not answer the call request at node 3, and node 3 will send the call request up the tree. The call request will reach database b, which again routes it back to node 3 if b has not yet received the address update message. The call request circulates back and forth between nodes b and 3 in this manner until the address update message reaches b, after which b will route the message to node 6, x's new address.

Clearly, the number of updates and accesses generated by movements and call set-up requests is determined by the particular placement of databases in the spanning tree. A large number of databases significantly increases the number of updates and accesses, which may create excessive delays. On the other hand, a small number of databases may not be sufficient to handle the number of updates and accesses generated in the network. We now formulate a combinatorial optimization problem that captures this tradeoff.

We assume that estimates are available for the rate of movements and calls between each pair of nodes in the network. According to our protocol, the path followed by messages due to a call from node i to a user at node j is the same as that followed by messages due to a movement from node j to node i. Thus, the databases that are accessed for a call from i to j are the same as those which would be updated for a movement from j to i. Let m_{ij} be the number of movements per unit time from node j to node i, and let c_{ij} be the number of calls per unit time originating at node i for users at node j. For simplicity, we assume that all calls and movements are between nodes within the network. Calls originating at node i for users at i do not require database accesses, so we set $c_{ii} = 0$ for all $i \in T$. If the cost of a database access is C_a, and the cost of a database update is C_u, the *transaction rate from i to j* is defined as

$$t_{ij} = C_u m_{ij} + C_a c_{ij}, \tag{1}$$

for all $i, j \in T$. The quantity t_{ij} represents the total cost per database due to access or update messages originating from node i and destined for node j. For disjoint subsets $R, S \subset T$, we also write $t(R \to S)$ for the transaction rate from R to S and $t(R \leftrightarrow S)$ for the transaction rate between R and S. Specifically,

$$t(R \to S) = \sum_{i \in R} \sum_{j \in S} t_{ij} \ ,$$

$$t(R \leftrightarrow S) = \sum_{i \in R} \sum_{j \in S} (t_{ij} + t_{ji}) \ ,$$

$$t(\{i\} \to \{j\}) = t_{ij} \ .$$

We will also refer to the *transaction rate at a database* b as the total cost incurred by b due to accesses and updates generated by all users in the network. This database transaction rate will be defined precisely in terms of the set of transaction rates $\{t_{ij}\}$.

We assume that each database can process no more than C transactions per unit time. The number C is referred to as the *database capacity*, and is not to be confused with storage capacity. Let A denote the set of nodes containing databases, or the set of *database sites*. Our optimization problem can be informally stated as follows:

> Choose A to minimize the sum of the transaction rates seen at all databases, subject to the constraint that the transaction rate at any database node in A does not exceed C.

In order to state this problem formally, we first introduce some more notation. Given a rooted tree T, node j is an *ancestor* of node k (or equivalently, node k is a *descendant* of node j) if j is in the unique path from k to the root. By convention, a node is its own ancestor and descendant. A *common ancestor* of nodes j and k is a node which is an ancestor of both j and k. The *least common ancestor* of nodes j and k, denoted as $\mathrm{LCA}(j,k)$, is the unique node which is a common ancestor of j and k but not an ancestor of any other common ancestor of j and k. Let $T(i)$ denote the set of nodes that are descendants of i.

For any node i, define

$$\alpha_i = \sum_{(j,k)\ :\ \mathrm{LCA}(j,k)=i} t_{jk}, \tag{2}$$

where the sum is over all distinct ordered pairs. That is, α_i is the sum of all transaction rates corresponding to signaling messages which originate in the subtree routed at i, and must pass through node i. Let $\{j \to k\}$ denote the set of nodes belonging to the path from node j to node k. By convention, $j, k \in \{j \to k\}$. For each node i, we define the quantity

$$\phi_i = t(T(i) \leftrightarrow T - T(i)) = \sum_{(j,k):i\in\{j\to k\},i\neq\mathrm{LCA}(j,k)} t_{jk}. \tag{3}$$

The quantity ϕ_i is therefore the sum of transaction rates due to messages traveling between the subtree routed at i and the rest of the tree. The quantities α_i and ϕ_i are illustrated in Figure 2. Note that $\alpha_i + \phi_i = \sum_{(j,k):i\in\{j\to k\}} t_{jk}$ is the minimum transaction rate corresponding to messages that must pass through node i. If node i is not a database site, then ϕ_i does not represent actual updates and accesses; however, updates and accesses represented by α_i must be processed at *some* ancestor of i. Note that the transaction rate at a database at node i is generally greater than $\phi_i + \alpha_i$, since there can be transaction rates associated with messages generated in the subtree routed at i which are not represented by α_i, but must be processed by i. These additional transaction rates will be defined shortly.

Proposition 1 *Let A be the set of nodes with databases. Then the sum of transaction rates at all databases is*

$$\lambda = \sum_{i\in T} \alpha_i + \sum_{i\in A} \phi_i. \tag{4}$$

Proof: Denote the lowest ancestor of LCA(j,k) which contains a database as a_{jk}. Note that if $a_{jk} \neq \text{LCA}(j,k)$, then there is no database at LCA(j,k) or at any node between LCA(j,k) and a_{jk}. A movement from j to k causes updates at all databases along the two paths from j to a_{jk} and from a_{jk} to k. By (2) and (3), the transaction rate at a_{jk} due to movements and calls between j and k is one term in $\alpha_{LCA(j,k)}$, and the transaction rate at any other database node i in the path $\{j \to k\}$ due to movements and calls between j and k is one term in ϕ_i. Summing over all pairs of nodes in the network then gives (4). □

Note that the first term in (4) is a constant independent of the choice of A, and therefore can be ignored in the following optimization. The cost of placing a database at node i is therefore ϕ_i.

We now specify the transaction rate at a database, which must be less than C. For nodes i and j, let $j \preceq i$ denote that j is a descendant of i, and let $j \prec i$ denote that j is a *proper* descendant of i (i.e., $j \preceq i$ and $j \neq i$). Given the set of database sites A, and a node i which is not a leaf, we define the subset of nodes

$$W_A(i) = \{j \preceq i : \nexists k \in A, j \preceq k \prec i\}. \tag{5}$$

This set, referred to as the *waffle* of A at i, contains all descendants j of i such that there is no database at j, and there are no databases between j and i. Note that $i \in W_A(i)$. If a database is placed at i, for every $j \in W_A(i)$ it must process all updates and accesses due to movements and calls between nodes for which j is the least common ancestor. The total transaction rate associated with these updates and accesses is α_j. The transaction rate for a database at node $i \in A$, denoted λ_i, is therefore

$$\lambda_i = \phi_i + \sum_{j \in W_A(i)} \alpha_j \leq C. \tag{6}$$

The capacity constraint (6) actually applies to all nodes $i \in T$, as we shall see shortly.

Proposition 2 *If $i \in T - A$ and k is i's parent, then*

$$\phi_k + \sum_{j \in W_A(k)} \alpha_j \geq \phi_i + \sum_{j \in W_A(i)} \alpha_j$$

.

Proof: By definition,

$$\phi_i = \sum_{(j,\ell): i \in \{j \to \ell\}, i \neq \text{LCA}(j,\ell)} t_{j\ell}. \tag{7}$$

For each (j,ℓ) in the summation range, we have $k \in \{j \to \ell\}$. Therefore,

$$\phi_i \leq \sum_{(j,\ell): k \in \{j \to \ell\}} t_{j\ell} = \phi_k + \alpha_k. \tag{8}$$

Combining this with the fact that $W_A(i) \cup \{k\} \subseteq W_A(k)$ gives

$$\phi_k + \sum_{j \in W_A(k)} \alpha_j \geq \phi_k + \alpha_k + \sum_{j \in W_A(i)} \alpha_j \geq \phi_i + \sum_{j \in W_A(i)} \alpha_j. \tag{9}$$

□

By repeated applications of Proposition 2, violation of (6) at a node $i \notin A$ implies violation of (6) at the lowest ancestor of i that contains a database. The constraint (6) therefore applies to *all* nodes $i \in T$.

The optimization problem can now be stated formally as

$$\begin{aligned} \text{Problem } \mathbf{P}: \qquad & \text{Minimize} \quad \sum_{j \in A} \phi_j & (10) \\ & \text{subject to} \quad A \subseteq T \\ & \qquad\qquad \phi_i + \sum_{j \in W_A(i)} \alpha_j \leq C, \text{ for all } i \in T & (11) \end{aligned}$$

Consider the inequality in Proposition 2. Its right-hand side is greater than or equal to $\phi_i + \alpha_i$; its left-hand side is less than or equal to C. Therefore a necessary condition for the existence of a feasible solution is $C \geq \max_{i \in T}(\alpha_i + \phi_i)$. This condition is also sufficient, since it ensures that a configuration with databases at every node is feasible. Furthermore, if $C \geq \sum_{i \in T} \alpha_i$, then a single database at the root of T is a solution to Problem **P**. The nontrivial range of C for our optimization problem is therefore $\max_{i \in T}(\alpha_i + \phi_i) \leq C < \sum_{i \in T} \alpha_i$.

3 Algorithm for Optimal Database Placement

Throughout this section we assume that T is fixed and given, and that α_i and ϕ_i are known at each node i. The basic idea is to consider a generalization of Problem **P**, which can be solved iteratively. In the following discussion we assume that α_i, ϕ_i, and C are specified as integers. Given rational quantities, or rational approximations to real-valued quantities, then this assumption can be satisfied by appropriately scaling these parameters. It is easily seen that the solution to Problem **P** is invariant to this scaling.

For each $i \in T$, and each integer θ, $0 \leq \theta \leq C$, we define Problem $\mathbf{P}(i, \theta)$ below. This problem asks for the optimal placement A' of databases in the subtree $T(i)$ which minimizes the transaction cost in this subtree subject to the capacity constraint and an additional constraint that the waffle at its root i be of "size" exactly θ.

$$\text{Problem } \mathbf{P}(i, \theta): \qquad \text{Minimize} \quad \sum_{\ell \in A'} \phi_\ell$$

$$\begin{aligned} \text{subject to} \quad & A' \subseteq T(i), \\ & \sum_{k \in W_{A'}(i)} \alpha_k = \theta, \\ & \phi_j + \sum_{k \in W_{A'}(j)} \alpha_k \leq C, \text{ for all } j \in T(i), \end{aligned}$$

provided $\theta > 0$; and

$$\begin{aligned} \text{Problem } \mathbf{P}(i,0): \qquad \text{Minimize} \quad & \sum_{\ell \in A'} \phi_\ell \\ \text{subject to} \quad & A' \subseteq T(i), \\ & i \in A, \\ & \phi_j + \sum_{k \in W_{A'}(j)} \alpha_k \leq C, \text{ for all } j \in T(i). \end{aligned}$$

A solution to $\mathbf{P}(i,\theta)$ consists of the minimum, denoted by $R^*(i,\theta)$, and a set of nodes $A' \subseteq T(i)$ which achieves this minimum, denoted by $A^*(i,\theta)$. Note that Problem $\mathbf{P}$ is precisely the same as Problem $\mathbf{P}(r,0)$, where r is the root. For a node $i \in T$, denote the collection of problems at i by

$$\mathbf{P}(i) \stackrel{def}{=} \{\mathbf{P}(i,\theta) : 0 \leq \theta \leq C\}.$$

We present an iterative, or dynamic programming, algorithm for solving $\mathbf{P}(r)$. Note it necessarily also solves $\mathbf{P} = \mathbf{P}(r,0)$. The algorithm can best be presented in its recursive form, for which a pseudo-code is shown in Table 1. Although the programming appears to be in top-down fashion, listing the root and parents before listing leaves and children; its excecution actually occurs in bottom-up fashion, solving for leaves and children nodes before glueing to obtain solutions for parents and the root. The key "glue" operation is described below.

Let $D(i)$ denote the set of nodes that are children of i. Having solved $\mathbf{P}(\ell)$ for all $\ell \in D(i)$, we "glue" these solutions to form a solution for $\mathbf{P}(i)$ as follows:

The Glue Operation: Compute

$$\begin{aligned} R^*(i,\theta) = \quad \text{Minimum} \quad & \sum_{\ell \in D(i)} R^*(\ell, \theta_\ell) \\ \text{subject to} \quad & 0 \leq \theta_\ell \leq C, \text{ for all } \ell \in D(i), \\ & \alpha_i + \sum_{\ell \in D(i)} \theta_\ell = \theta, \end{aligned} \tag{12}$$

for each θ, $1 \leq \theta \leq C$; and compute

$$R^*(i,0) = \quad \text{Minimum} \quad \phi_i + \sum_{\ell \in D(i)} R^*(\ell, \theta_\ell) \tag{13}$$
$$\text{subject to} \quad 0 \leq \theta_\ell \leq C, \text{ for all } \ell \in D(i),$$
$$\phi_i + \alpha_i + \sum_{\ell \in D(i)} \theta_\ell \leq C.$$

Also compute

$$A^*(i,\theta) = \cup_{\ell \in D(i)} A^*(\ell, \theta^*_\ell)$$

for each θ, $1 \leq \theta \leq C$; and compute

$$A^*(i,0) = \{i\} \cup_{\ell \in D(i)} A^*(\ell, \theta^*_l),$$

where $\{\theta^*_\ell, \ell \in D(i)\}$ is a set of minimizing values in (12) (or (13)). By convention, $R^*(i,\theta) = \infty$ if the minimization range in (12) (or (13)) is an empty set.

The intuition behind the algorithm is as follows. Given a feasible solution to Problem **P**, i.e., a set of databases A, the associated *resource accumulation* at a node i is defined as $R(i) = \sum_{\ell \in A(i)} \phi_\ell$, where $A(i) = A \cap T(i)$ is the set of databases in the subtree rooted at i. The quantity $R(i)$ is the accumulation of database costs in $T(i)$. The associated *constraint accumulation* at node i is defined as $\theta(i) = \sum_{k \in W_{A(i)}(i)} \alpha_k$. It is the accumulation of transaction rates which applies towards the capacity contraint at the parent node of i (or the capacity constraint at i itself if $\theta(i) = 0$). Solving $\mathbf{P}(i)$, therefore, can be accomplished by solving $\mathbf{P}(\ell)$ for all chidren ℓ of i and then "glueing" the solutions together.

The complexity of the algorithm is determined by that of the glueing operations. Consider a node i for which $D(i)$=$\{1,2\}$. For each $\theta \neq 0$ there are at most C values of (θ_1, θ_2) involved in the minimization (12). For $\theta = 0$, there are at most C^2 values of (θ_1, θ_2) in the minimization (13). The overall complexity of the minimizations involved for $0 \leq \theta \leq C$ is therefore $O(C^2)$. More generally, let $d(i) = |D(i)|$ denote the number of children, or the *degree*, of i. If $d(i) > 2$, the subsolutions for the first two children, i_1 and i_2, can be combined using the following rule:

$$R^*((i_1,i_2);\theta) = \min \{R^*(i_1,\theta_1) + R^*(i_2,\theta_2) : \ 0 \leq \theta_1, \theta_2 \leq C \text{ and } \theta_1 + \theta_2 = \theta\},$$

$$A^*_{(i_1,i_2)}(\theta) = A^*_{i_1}(\theta^*_1) \cup A^*_{i_2}(\theta^*_2),$$

where $0 \leq \theta \leq C$, and where θ^*_1, θ^*_2 achieves the minimum. Reasoning as before, the complexity of this operation (summing over the computations for $0 \leq \theta \leq C$) is $O(C^2)$, and the resulting solution can be viewed as the subsolution for a single "node" (i_1, i_2) which replaces

```
main{}
    {
        solve(P(r));
    }
subroutine solve (P(i))
    {
        if i is a leaf, set R*(i, α_i) = 0 and A*(i, θ) = ∅ for every θ, 0 ≤ θ ≤ C.
        else
            {
                for each ℓ ∈ D(i), solve(P(ℓ));
                "glue" solutions from all children;
            }
    }
```

Table 1: Recursive pseudo-code for the algorithm

the individual nodes i_1 and i_2. Iterating this procedure $d(i) - 2$ times effectively reduces the number of children to two, at which point, as described earlier, an $O(C^2)$ computation gives the desired subsolution for i by glueing these two children. The complexity of combining the subsolutions for the children of i is therefore $O(d(i)C^2)$. Summing over the nodes of the tree, and using the fact that $\sum_{i \in T} d(i) = |T| - 1$, where $|T|$ is the number of nodes in T, we obtain an overall complexity of $O(|T|C^2)$.

The complexity of the algorithm clearly depends on the particular integer value of C which is chosen. This is determined by the precision with which the parameters α_i, ϕ_i, and C are initially specified. If these parameters are coarsely quantized, then the scale factor that converts these parameters to integers need not be large. This is likely to be the case in practice, since in most situations mobility and call rates cannot be specified with great precision.

The dynamic programming solution presented here also applies for real valued α_i, ϕ_i, and C. However, in this case, since the precision is not truncated, solutions must be preserved for all possible $\theta \leq C$, rather than just for integer θ. The size of the subsolutions, and hence the complexity of the algorithm, may grow exponentially with the number of nodes. Whether or not a solution to Problem $\mathbf{P}$ exists in this case with complexity which is polynomial in $|T|$ (and independent of C) is an open problem.

4 Example

Consider a square region $\mathcal{S}$ which is subdivided into 16 smaller squares, as shown in Figure 3. Each small square, referred to as a *location area*, represents a geographical area in which a user may be located, and is associated with a leaf of the signaling network T. Each internal node of T in this example represents a switching node which may contain a database. In practice, the leaves of T may be gateways into networks covering each of the smaller squares.

Construction of T is illustrated in Figure 4. Assigning a switching node to each of the boundaries shown in Figure 4, and to analogous boundaries in the remaining parts of $\mathcal{S}$, results in the tree shown in Figure 5. Nodes associated with boundaries that create congruent partitions are labelled by the same letter. The root node a corresponds to the boundary AA, the node b to boundary BB, c to CC, d to DD, and e to DE.

Associated with each node $i \in T$ are the parameters (α_i, ϕ_i). In this example, these values are determined using a fluid flow model of mobility in which the rate of movements between two regions is the same as the length of their common boundary. For simplicity, we ignore the contribution to the transaction rates t_{ij} due to database accesses. If the length of the side of a small square is one, then the mobility rate between location areas 1 and 2 is $m_{12} + m_{21} = 1$. The mobility rate between location areas 1 and 3 equals zero since they have no common boundary. To compute the quantities α_i and ϕ_i for each node $i \in T$, we first note that i corresponds to a boundary $\mathcal{B}$ which divides a region $\mathcal{R}$ into two sections. ¿From (2)-(3), it is easy to see that α_i equals the length of $\mathcal{B}$ and ϕ_i equals the length of the common boundary between $\mathcal{R}$ and $\mathcal{S} - \mathcal{R}$. The pairs (α_i, ϕ_i) corresponding to some of the nodes in T are shown in Figure 5.

As explained in the previous section, the nontrivial range of C is $6 = \max_{i \in T}(\alpha_i + \phi_i) \leq C < \sum_{i \in T} \alpha_i = 24$. In this example we take $C = 12$, so that the databases have one half of the capacity required for a centralized scheme.

For each node i we can list solutions to Problem $\mathbf{P}(i, \theta)$ for $\theta = 0, \ldots, C$ in a table $S(i)$. Each entry consists of θ, $R^*(i, \theta)$, and $A^*(i, \theta)$. Values of θ for which $R^*(i, \theta) = \infty$ are omitted from the table, since they are infeasible.

Referring to Figure 5, we start by computing solutions to Problem $\mathbf{P}(d, \theta)$ and Problem $\mathbf{P}(e, \theta)$. The set $A(d)$ can be one of only two possibilities: either $A(d) = d$ or $A(d)$ is empty. The former case corresponds to $\theta(d) = 0$, and since $\alpha_d + \phi_d = 1 + 3 \leq C = 12$, this is feasible. The resource accumulation is $\phi_d = 3$. If $A(d)$ is empty, then $\theta = \alpha_d = 1$, and there is zero resource accumulation. The solutions for node e are generated similarly. We therefore have the following tables:

$S(d)$

θ	$R^*(d, \theta)$	$A^*(d, \theta)$
0	3	$\{d\}$
1	0	$\emptyset$

$S(e)$

θ	$R^*(e, \theta)$	$A^*(e, \theta)$
0	5	$\{e\}$
1	0	$\emptyset$

These subsolutions can now be glued together to produce the following table $S(c)$. The

table $S(c_1)$ is also shown, and is obtained from $S(c)$ by symmetry.

$S(c)$

θ	$R^*(c,\theta)$	$A^*(c,\theta)$
0	4	$\{c\}$
$\star 2$	8	$\{d,e\}$
3	3	$\{d\}$
4	0	$\emptyset$

$S(c_1)$

θ	$R^*(c_1,\theta)$	$A^*(c_1,\theta)$
0	4	$\{c_1\}$
$\star 2$	8	$\{d_1,e_1\}$
3	3	$\{d_1\}$
4	0	$\emptyset$

It is usually the case that a higher constraint accumulation θ reduces the associated resource accumulation. However, all table entries marked by asterisks correspond to instances for which there exists a $\theta^* < \theta$ such that $R^*(i,\theta^*) \leq R^*(i,\theta)$. In this case, the solution to Problem $\mathbf{P}(i,\theta)$ cannot be part of the solution to Problem $\mathbf{P}$, since the solution to Problem $\mathbf{P}(i,\theta^*)$ yields a lower cost. The corresponding entry can therefore be deleted from $S(i)$ when performing the glue operation.

The solutions for nodes c and c_1 can now be glued together, and this gives $S(b)$. In general, there are several choices for $A^*(b,\theta)$. For each θ we list one representative choice, and note that the remaining choices can be easily deduced by symmetry. As before, the table $S(b_1)$ can be easily obtained from $S(b)$.

$S(b)$

θ	$R^*(b,\theta)$	$A^*(b,\theta)$
0	8	$\{b,c\}$
$\star 2$	8	$\{c,c_1\}$
5	7	$\{c,d_1\}$
6	4	$\{c\}$
$\star 8$	6	$\{d,d_1\}$
9	3	$\{d\}$
10	0	$\emptyset$

$S(b_1)$

θ	$R^*(b_1,\theta)$	$A^*(b_1,\theta)$
0	8	$\{b_1,c_2\}$
$\star 2$	8	$\{c_2,c_3\}$
5	7	$\{c_2,d_3\}$
6	4	$\{c_2\}$
$\star 8$	6	$\{d_2,d_3\}$
9	3	$\{d_2\}$
10	0	$\emptyset$

Since Problem $\mathbf{P}$ is identical to Problem $\mathbf{P}(a,0)$, the solution to Problem $\mathbf{P}$ can be obtained by glueing together the tables $S(b)$ and $S(b_1)$. We obtain

$$R^*(a,0) = 12, \quad A^*(a,0) = \{a,b,c,c_2\}$$

.
Of course, there are other symmetrically equivalent choices for $A^*(a,0)$.

The set $A^*(a,0)$ is illustrated in Figure 6. If the signaling network T, as well as the set of database nodes within T, is subject to design, then T can be simplified by eliminating switching nodes without databases and their associated links. This is illustrated in Figure

6, which shows a simplified signaling network obtained from the original T with optimally placed database nodes. The simplified network retains only those nodes of T with databases along with the leaves of T and links required to connect the nodes. This configuration therefore gives a grouping of the leaves and a placement of databases which takes into account geographical constraints and user density as well as the available database capacity.

5 Generalizations

The dynamic programming algorithm described in Section 3 can of course be used to solve any problem that has the structure of Problem **P**, stated at the end of Section 2. Furthermore, the idea of augmenting the problem by considering constraint and resource accumulations and working up recursively from the leaves to the root of the tree can also be used to address somewhat more general formulations. In this section we discuss a natural queueing theoretic formulation of the problem of designing an optimal signalling hierarchy that is amenable to this treatment.

Considering the signalling tree T as before, and with $P(i)$ denoting the parent of node i, let i^u and i^d denote the directional links from i to $P(i)$ and from $P(i)$ to i, respectively. The up link i^u carries traffic up from i, while the down link i^d carries traffic down to i. Let us now model the communication links and databases as queues. We refer to the links and databases together as network elements. For simplicity, assume that for each network element the service time distributions for accesses and updates are the same. (Different network elements may have different service time distributions, however). Let $D_\nu(\lambda_\nu)$ denote the average delay at the element ν, where $\lambda_\nu = \lambda_\nu^a + \lambda_\nu^u$ is the transaction rate seen by the element ν, which is the sum of the rate of accesses λ_ν^a and the rate of updates λ_ν^u. This assumes that each element can be viewed in isolation from the point of view of its average delay, and is a well-accepted approximation justified in a variety of ways, including the Kleinrock independence approximation and Jackson's theorem for networks of M/M/1 queues (see Sections 3.6-3.8 and 5.4 of [3]). One can see (p. 165 of [3] , for instance) that the average delay incurred by a call set-up request is proportional to $\sum_\nu \lambda_\nu D_\nu(\lambda_\nu)$, where we sum over all elements of the network. The cost of element ν can therefore be taken to be $K_\nu(\lambda_\nu) = \lambda_\nu D_\nu(\lambda_\nu)$. Note that the input rate at each element is completely specified if we know the set of databases A, and the transaction rates $t_{jk} = m_{jk} + c_{jk}$ for all nodes j and k.

In particular consider modelling the processes of updates and accesses as independent Poisson processes and model the databases and communication links as M/M/1 queues which serve messages at rates μ_d and μ_c, respectively. Assume, for simplicity, that the switching cost is zero, so that nodes that do not contain databases have zero cost. The average delay for an M/M/1 queue with input rate λ and service rate μ is given by $1/(\mu - \lambda)$ (see p. 129 of [3]), where the implicit capacity constraint $\lambda < \mu$ is necessary and sufficient for stability,

i.e., finite delay. Then, for an input rate λ, the cost of a database is

$$K_d(\lambda) \;=\; \begin{cases} \lambda/(\mu_d - \lambda), & \lambda \;<\; \mu_d, \\ \infty, & \text{otherwise.} \end{cases} \tag{14}$$

and that of a communication link is

$$K_c(\lambda) \;=\; \begin{cases} \lambda/(\mu_c - \lambda), & \lambda \;<\; \mu_c, \\ \infty, & \text{otherwise.} \end{cases} \tag{15}$$

The input rate λ_i that would be seen by a database at node i is given, as in Section 2, by

$$\lambda_i \;=\; \phi_i + \sum_{j \in W_A(i)} \alpha_j, \tag{16}$$

This determines the contribution to the cost of placing a database at node i. It remains to determine the input rates for the communication links, and hence the link costs. This depends on which one of two interpretations is adopted for the links. In the first, the communication links are viewed as *physical* links of given capacity, and in the second, the links are *virtual*, in the sense that the link costs must reflect the fact that nodes which do not contain databases, along with the incident links, are eliminated. (See Figure 6 for an illustration.)

Let $\phi_i^u = t(T(i) \to T - T(i))$ be the traffic on i^u and $\phi_i^d = T(T(i) \leftarrow T - T(i))$ be the traffic on i^d. Then $\phi_i^u + \phi_i^d$ equals the previously defined quantity ϕ_i. For physical links, transactions corresponding to a constraint accumulation θ at node i cause traffic at rate θ in both directions between i and its lowest ancestral database. Thus, the net traffic λ_{i^u} on the upward link i^u is $\theta + \phi_i^u$ and the net traffic λ_{i^d} on the downward link i^d is $\theta + \phi_i^d$. For virtual links, the links i^u and i^d actually exist only if a database is placed at i. Thus, since a constraint accumulation $\theta > 0$ at node i implies that node i does not contain a database, no communication cost is incurred on the links i^u and i^d, since these are virtual links which will be eliminated. Thus, we can set $\lambda_{i^u} = \lambda_{i^d} = 0$ if $\theta_i > 0$. On the other hand, for $\theta_i = 0$, which corresponds to placing a database at i, we get $\lambda_{i^u} = \phi_i^u$ and $\lambda_{i^d} = \phi_i^d$.

In either of the two situations above, our objective is to minimize

$$\sum_{i \in A} K_d(\lambda_i) + \sum_{i \in V} [K_c(\lambda_{i^d}) + K_c(\lambda_{i^u})], \tag{17}$$

where, as described above, the input rates λ_ν for each element ν depends on the problem parameters and on the choice of A. The constraint accumulation $\theta(i)$ is as in Section 3. The resource accumulation $R(i)$ at i is given by

$$R(i) \;=\; \sum_{j \in A \cap T(i)} K_d(\lambda_j) + \sum_{j \in T(i)} [K_c(\lambda_{j^d}) + K_c(\lambda_{j^u})]. \tag{18}$$

The definition of the minimum resource accumulation $R^*(i, \theta)$ is as in Section 3. We can now write down the glueing equations which specify the dynamic programming algorithm. We again assume that α_i, ϕ_i, and μ_d are integers.

Physical Link: For $\theta = 1, \cdots, \mu_d - 1$, compute

$$\begin{aligned} R^*(i,\theta) = \quad & \text{Minimum} \quad K_c(\theta + \phi_i^u) + K_c(\theta + \phi_i^d) + \sum_{\ell \in D(i)} R^*(\ell, \theta_\ell) \\ & \text{subject to} \quad 0 \le \theta_\ell \le C, \text{ for all } \ell \in D(i), \\ & \qquad\qquad \alpha_i + \sum_{\ell \in D(i)} \theta_\ell = \theta, \end{aligned}$$

and for $\theta = 0$ compute

$$\begin{aligned} R^*(i,0) = \quad & \text{Minimum} \quad K_d(\phi_i + \alpha_i + \sum_{\ell \in D(i)} \theta_\ell) + K_c(\phi_i^u) + K_c(\phi_i^d) + \sum_{\ell \in D(i)} R^*(\ell, \theta_\ell) \\ & \text{subject to} \quad 0 \le \theta_\ell \le C, \text{ for all } \ell \in D(i), \\ & \qquad\qquad \phi_i + \alpha_i + \sum_{\ell \in D(i)} \theta_\ell < \mu_d. \end{aligned}$$

Also compute

$$A^*(i,\theta) = \cup_{\ell \in D(i)} A^*(\ell, \theta_\ell^*)$$

if $\theta > 0$, and compute

$$A^*(i,0) = \{i\} \cup_{\ell \in D(i)} A^*(\ell, \theta_\ell^*),$$

where $\theta_\ell^*, \ell \in D(i)$ are minimizing arguments.

Virtual Link: For $\theta = 1, \cdots, \mu_d - 1$, compute

$$\begin{aligned} R^*(i,\theta) = \quad & \text{Minimum} \quad \sum_{\ell \in D(i)} R^*(\ell, \theta_\ell) \\ & \text{subject to} \quad 0 \le \theta_\ell \le C, \text{ for all } \ell \in D(i), \\ & \qquad\qquad \alpha_i + \sum_{\ell \in D(i)} \theta_\ell = \theta, \end{aligned}$$

and for $\theta = 0$ compute

$$\begin{aligned} R^*(i,0) = \quad & \text{Minimum} \quad K_d(\phi_i + \alpha_i + \sum_{\ell \in D(i)} \theta_\ell) + K_c(\phi_i^u) + K_c(\phi_i^d) + \sum_{\ell \in D(i)} R^*(\ell, \theta_\ell) \\ & \text{subject to} \quad 0 \le \theta_\ell \le C, \text{ for all } \ell \in D(i), \\ & \qquad\qquad \phi_i + \alpha_i + \sum_{\ell \in D(i)} \theta_\ell < \mu_d. \end{aligned}$$

Also compute $A_i^*(\theta)$ as before for the Physical Links Case.

Consider once again the example of Section 4, and assume that $\phi_i^u = \phi_i^d = \phi_i/2$. We assume that the databases are M/M/1 queues with service rate $\mu_d = 12$. We consider three distinct situations:
(A) No communication costs: Communication links have infinite speed ($\mu_c = \infty$ and links may be interpreted as either physical or virtual);
(B) Physical links with service rate $\mu_c = 24$ in each direction;
(C) Virtual links with service rate $\mu_c = 24$ in each direction.

Since the glueing procedure has already been illustrated in Section 4, we omit all details and provide the results of the optimization in Figures 7 and 8 (as before, only one representative of each set of symmetrically equivalent solutions is shown). Since the delay in M/M/1 queues increases sharply with the utilization, the number of databases in the optimized set A^* required in case (A) above is larger than that for the linear constraint considered in Section 4 (compare Figure 7 with Figure 6). As shown in Figure 7, there are two different database configurations achieving the optimization in case (A). (That is, the same average delay is incurred whether or not there is a database at node b). Since the communication cost is zero for $\mu_c = \infty$, the links may be thought of as either physical or virtual. However, if we now introduce nonzero communication cost using physical links of capacity $\mu_c = 24$, *not* adding a database becomes more costly than for case (A). As shown in Figure 7, this resolves the tie in the two different optimal solutions for case (A) in favor of the solution with the extra database at node b. On the other hand, if the links are virtual, *adding* a database is more costly than for case (A). As shown in Figure 8, the optimal database hierarchy for case (C) has fewer databases than the solutions for cases (A) and (B).

6 Conclusions

We have proposed a hierarchical signaling scheme for mobility tracking in personal communications and have given a dynamic programming algorithm for determining the optimal placement of databases in the hierarchy. We illustrated how the dynamic programming algorithm can be used for optimizing a variety of objective functions. An important area for future research is the comparison of our hierarchical scheme with other methods for mobility tracking, such as assigning each user to a home location, as in GSM. It may be possible to design a hybrid scheme combining the best features of both methods. Additional topics include the design of signalling networks which can adapt to time-varying call and mobility traffic, and the optimization of the wireless resources used for mobility tracking.

7 Acknowledgements

We thank Ken Steiglitz and Seshadri Mohan for their helpful comments concerning this work, and Ravi Jain and Rich Wolff for comments on an earlier version of this paper. We

also thank Ilan Kessler for supplying a preprint of [2]. Techniques similar to those used in [2] enabled us to improve our estimate of the complexity of the dynamic programming algorithm.

References

[1] B. Awerbuch and D. Peleg, "Concurrent online tracking of mobile users," *Proc. ACM SIGCOMM Symposium on Communication, Architectures and Protocols*, October 1991.

[2] A. Bar-Noy and I. Kessler, "Tracking mobile users in wireless communications networks," IBM Research Report RC 18276 (80149), August 27, 1992 (also submitted for external publication).

[3] D. Bertsekas and R. Gallager, *Data Networks*. Prentice-Hall, 1987.

[4] T. Imielinski and B. R. Badrinath, "Querying in highly distributed mobile environments," *Proc. 18th VLDB Conf.*, Vancouver, British Columbia, Canada, 1992.

[5] J. A. Audestad, "GSM general overview of network functions," *Proc. Int. Conf. Digital Land Mobile Radio Commun.*, Venice, 1987.

[6] K. S. Meier-Hellstern, E. Alonso, and D. R. O'Neil, "The use of SS7 and GSM to support high density personal communications," *Proc. ICC '92*, pp. 356.2.1-356.2.5.

[7] C. N. Lo, R. S. Wolff, and R. C. Bernhardt, "Expected network database transaction volume to support voice personal communications services," Bellcore Technical Memorandum TM-ARH-020480, Feb. 5, 1992.

[8] C. N. Lo, S. Mohan, and R. S. Wolff, "Performance modeling and simulation of data management for personal communications applications," to appear, *Proc. 2nd Bellcore Symp. Perf. Modeling*, 1992.

[9] C. Wang, J. Wang, and Y. Fan, "Performance evaluation of a hierarchical location registration and call routing for personal communications," *Proc. ICC '92*, pp. 356.6.1-356.6.4.

[10] J. Z. Wang, "The hierarchical structure of tracing strategy for universal personal communication systems," *Proc. UPC '92*, pp. 09.04.1-09.04.5.