Wireless Networks 7, 497–512, 2001
© 2001 Kluwer Academic Publishers. Manufactured in The Netherlands.

Load-Balanced Location Management for Cellular Mobile Systems Using Quorums and Dynamic Hashing

RAVI PRAKASH *
Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688, USA

ZYGMUNT HAAS **
School of Electrical Engineering, Cornell University, Ithaca, NY 14853, USA

MUKESH SINGHAL
Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210, USA

Abstract. This paper presents a new distributed location management strategy for cellular mobile systems. Its salient features are fast location update and query, load balancing among location servers, and scalability. The strategy employs dynamic hashing techniques and quorums to manage location update and query operations. The proposed strategy does not require a home location register (HLR) to be associated with each mobile node. Location updates and queries for a mobile node are multicast to subsets of location servers, which change with time depending on the location of mobile node/querying node and load on the servers. Simulation experiments demonstrate that such dynamism prevents situations of heavy load on some location servers when mobile nodes are not uniformly distributed in space, or when some mobile nodes have their location updated or queried more often than others. Also, queries return the latest location information of a mobile node with a very high probability. The proposed scheme requires at most one unicast and two rounds of message multicasting for location update and query operations. All multicast messages have a small payload and are restricted to the high bandwidth wired part of the mobile network resulting in low communication overhead. Moreover, if a quorum of location servers gets overloaded, part of its load can be transferred to another lightly loaded quorum using dynamic hashing techniques.

Keywords: mobile computing, distributed location management, dynamic hashing, tries, quorum systems, location independent numbering

1. Introduction

In cellular mobile computing systems the service area is divided into several registration areas (RAs), with each RA composed of one or more neighboring cells. Location management pertains to operations for updating and retrieving location information about mobile nodes at the granularity of RA. The ability of mobile nodes (MNs) to autonomously move from one part of the network to another raises interesting issues in the management of location information of these nodes.

1. *Frequency of location updates.* Location updates for an MN should be performed when the MN moves out of one RA into another. Three alternatives, namely time-based, movement-based, and distance-based strategies to determine update intervals have been proposed in [6].
2. *Information organization.* As centralized location management strategies are neither robust, nor scalable, we will concentrate on distributed solutions. Several strategies for hierarchical organization of location information have been proposed, for example, [4,28].
3. *Information placement.* Should the set of location servers associated with a mobile node be dependent solely on the identity of the mobile node, or also on the location of the mobile node whose information is being updated? How many copies of location information for a mobile node should be maintained in the system? Larger degrees of replication may lead to quick retrieval of information while incurring high storage and synchronization overheads.
4. *Information retrieval.* Who should be probed to obtain an MN's location information? Assuming there are multiple location servers in the system, should the set of queried servers depend on (i) the identity of the MN, (ii) location of the node that is trying to locate the MN? The goal should be to quickly retrieve location information and incur low communication overheads.
5. *Paging.* Once location information has been retrieved at RA granularity, how is the exact cell location of the MN determined? Several paging schemes have been proposed to determine the cell, within the RA, in which the MN is present [3,26].
6. *Adaptibility.* The solution should be able to adapt to the arrival (departure) of mobile nodes into (from) the system, and significant changes in the update and query

* The author's work was supported in part by the National Science Foundation grant number CCR-9796331.
** The author's work was supported in part by the National Science Foundation grant number ANI-9805094.

rates of mobile nodes. Also, while the system is adjusting to fluctuations in load offered by a group of mobile nodes, location services (update and query) for other nodes should not be suspended.

7. *Local optimizations.* Several optimizations like caching location information [15], forwarding pointers [16] have been proposed to reduce the number of location queries.

8. *Suseptability to database failures.* Failures of databases that store location information may significantly affect the performance of a location management system [13]. Replication of the information, such as in the quorum-based scheme proposed here, reduces the effect of database failures, as compared with the current centralized schemes that do not employ replication.

It is extremely important that location management schemes adequately address the issues mentioned above. Otherwise, high communication overheads and/or delays will be experienced. Also, it should be possible to combine optimization schemes like caching and forwarding pointers with the update and query strategies to further improve the latter's performance.

We expect two new issues to gain importance with the increasing popularity of mobile systems:

1. *Location independent numbering.* Users would like to have numbers assigned to them "for life" without having to change them as they move from one place to another. Location management schemes based on standards like IS-41 [10] and GSM MAP [20] that rely on *home location register* (HLR) and *visitor location register* (VLR) based schemes may not be ideal in such a situation as the home location of a mobile node may change with time. Hence, new strategies need to be developed that discard the notion of HLRs and VLRs.

1. *Load balancing among location servers.* Some location servers may be overloaded with location update and query requests while other servers are relatively idle. Even though load balancing among servers has not been addressed in existing literature, due to its potential importance in future generations of mobile systems we feel solutions should be designed *a priori*. Note that sometimes load balancing may succeed in reducing the maximum load among servers at the cost of increased average load on all servers and/or increased communication over the network. The solution described in this paper exhibits such a behavior. Solutions that can minimize these overheads need further investigation.

In this paper we address the location *information organization*, *placement*, and *retrieval* issues described above. Specifically, we present a distributed location management scheme in which location information of mobile nodes (at the granularity of RA) is replicated at $O(\sqrt{N})$ location servers, where N is the total number of location servers in the system. A quorum based scheme [23] is employed to determine the set of servers to be targeted for location updates and queries. As a result, location updates and queries can be performed in $O(1)$ time, where the unit of time is the round-trip message propagation delay. This is a significant improvement over hierarchical tree-based schemes that incur $O(\lg N)$ round-trip message delays for location query in the worst case. In the best case scenario, when location information of a queried mobile node is available at a local server, both the proposed scheme and the hierarchical scheme have comparable performance. Later, we also describe how to use another replication strategy where location information of most mobile nodes is replicated at $O(\lg N)$[1] location servers. Hence, the storage overheads are comparable with those of hierarchical schemes as described in [4,12]. Also, the notion of HLRs and VLRs is discarded.

In fairness to hierarchical schemes, if location information about a queried mobile node is not available at a local server, location queries in the proposed quorum based scheme require multicasting to a group of $O(\lg N)$ servers over the fixed wireline network: a high communication overhead. However, as these messages are identical, they can be pushed as a single message towards a group of servers and split into identical copies close to their destination, i.e., multicast can be employed.

The proposed solution satisfies the *adaptibility* requirement through load-balancing among location servers. Load-balancing is achieved through dynamic hashing. *Optimization* schemes like caching location information [15], maintaining forwarding pointers [16], and search-updating [19] can be easily combined with the proposed solution to further improve its performance. Also, the proposed solution can work with most strategies for *paging* and for determining the *frequency of location updates*.

2. System model

We assume a wide area cellular system. Each cell has a base station, henceforth referred to as the *mobile service station* (MSS). A cell is uniquely identified by referring to its *MSS_id*. The MSSs are connected to each other by a fixed wireline network which also contains various location servers. An MSS can be in wireless communication with the mobile nodes (MNs) in its cell.

An MN communicates with other units, mobile or static, only through the MSS of the cell in which it is present. If a node (static or mobile) wishes to communicate with an MN, first it has to determine the location of the MN: the cell in which the MN is present.

The cellular coverage area is divided into several *registration areas* (RAs). Typically, a cluster of neighboring cells constitute an RA. A mobile node's location information is updated when it moves from one RA to another. This location information is stored at location server(s). A location query operation returns the identity of the RA, henceforth referred to as *RA_id*, in which the queried MN is present.

[1] $\lg N$ denotes $\log_2 N$.

Once the *RA_id* of the MN has been determined, paging is performed within the RA to determine the cell in which the MN is present.

We assume that one or more RAs constitute a *zone*. Each zone has a location server associated with it. The location server for a zone stores location information about every MN currently present in the zone, and some MNs outside the zone. The identity of MNs outside the zone whose location information is stored at a location server is determined by the proposed location update strategy. There are additional servers, also in the fixed wireline part of the network, that can be used as location servers to share the burden under high load situations. In low load situations, such servers can be relieved of their location management responsibilities and may perform other services.

3. Previous work

The IS-41 [10] and GSM MAP [20] standards for cellular radio communication use centralized location management schemes. Two kinds of servers, namely *Home Location Registers* (HLRs) and *Visitor Location Registers* (VLRs) are used. IS-41 associates an HLR with each MN. When an MN moves from one RA to another the MN registers its location with the VLR of the RA it enters. If the old and new RAs are associated with the same VLR, no further action is required. Otherwise, the new VLR informs the HLR that it will now be serving the MN. The HLR deregisters the MN at the old VLR and associates the MN with the new VLR. For location query, the querying node can identify the HLR to be queried based on the identity of the MN it is trying to locate. The HLR forwards the query to the VLR where the queried MN is known to be registered.

Various improvements have been suggested.[2] To reduce the signaling traffic due to location registration, *forwarding* [4,16] and *local anchoring* [14] have been proposed. Both these schemes reduce the location update overheads, but may incur extra delays during location queries. Depending on the call-to-mobility ratio, the two approaches lead to varying degrees of performance improvement.

In [27], a user profile replication scheme is proposed. A central processor collects information about the mobility and calling patterns for all the MNs. Based on this data, decisions are made about replicating location information (and other user profile information) about a mobile node at multiple location servers. Such a centralized scheme is not scalable as the analysis of data about all the MNs is computation-intensive and time-consuming [2]. Hence, distributed profile replication schemes are needed.

Distributed location management schemes [4,25,28] employ hierarchical databases to store location information about subsets of MNs. In [4], a hierarchy of distributed regional directories is maintained. The ith level regional directory enables a node, static or mobile, to track any mobile node within a distance of 2^i from it. Corresponding to each level i, *query* and *update* sets of directories are associated with nodes u, v such that $query_i(u) \cap update_i(v) \neq \emptyset$, $\forall u, v$ within 2^i distance from each other. The *update* set for a node is the set of directories where the location information of the node is stored. The *query* set for a target node is the set of directories that will be probed to find the location of the target node. If location information is not found at the ith level regional directories, the region of search is expanded by probing higher level directories. So, in [4], multiple rounds of probes may be required. If an MN is highly mobile, the time to locate it will be greater.

The locality of reference patterns is exploited in [25]. Nodes in an MN's *working set* communicate with the MN more frequently than nodes that are not in the working set. An MN can dynamically determine its working set depending on the call-to-mobility ratio between network node and MN pairs. Nodes in the working set are informed about the location update when an MN moves, while other nodes are made to search for the MN when they wish to communicate with the MN.

In [17], the issue of location independent numbering is addressed. As a VLR cannot determine the HLR for an MN solely from the MN's number, a mapping from the number to HLR needs to be stored. However, replicating such a mapping table at all the VLR is not practical. So, a new class of *translation servers* (TS), mapping an MN's number to its HLR, is introduced. Let a VLR need to determine an MN's HLR. The VLR employs a hash function to determine the TS that stores the appropriate mapping. The TS so identified is queried to determine the HLR for the MN. Thus, an extra step of table look-up and an extra set of translation servers are introduced.

In this paper we assume that a policy to determine the size and layout of registration areas (RAs) has already been decided. We proceed from that point and address the *information organization, placement, and retrieval* issues. We propose an algorithm for efficient location updates and queries. Also, once the RA for the queried MN has been determined, any paging strategy can be employed to pin-point the MN's cell. Location independent numbering is handled using quorums in conjunction with dynamic hashing. However, unlike [17], the proposed algorithm does not introduce an extra step for number to TS translation, and it does not require an extra set of *translation servers*. The hash function directly maps the location update or query operation for an MN to a small set of location servers.

4. Motivation and the basic idea

In several existing schemes, even though a mobile node is free to roam throughout the coverage area and still be accessible, its location information is, in a sense, immobile as all location registrations have to be conveyed to the HLR whose location is fixed. The distributed schemes proposed in the past suffer from high latency of location queries as

[2] A thorough survey of recent work in location management can be found in [2].

they may have to perform multiple directory look-ups in a sequential fashion, or follow a series of forwarding pointers. We propose a novel approach which is motivated by the two shortcomings mentioned above. Its salient features are:

1. The concept of HLRs is discarded. As an MN moves from one RA to another, the set of location servers storing its location information changes. Also, location queries originating in different parts of the system, for the same mobile node, are targeted towards different sets of location servers. As a result, MNs that are generating a large number of location updates, and/or are being queried very frequently do not lead to some location servers being overloaded.

2. In existing distributed schemes, like [4], concurrent querying is performed at the *query set* for a given level. However, the query rises through the levels *sequentially*. In the proposed scheme, if location information about the queried mobile node is not available at the location server for the local zone, a small set of location servers are queried concurrently. Similarly, location updates are sent concurrently to a set of location servers. Updates and queries can be accomplished in one or two round-trip message propagation times. If one waits until all responses are received, the slowest location server dominates the communication time. However, even then one would expect this to result in smaller delays than that incurred in multiple round-trip propagation times: one per level of the hierarchy until the information is retrieved.

4.1. Using quorums for fast update and query

We employ ideas discussed by Mullender and Vitányi for distributed match-making [21]. The proposed approach has the following characteristics: (i) multicast location updates for a mobile node to a set of location servers (update set), (ii) if location information of a mobile node is not available locally, multicast queries for the mobile node to a set of servers (query set), (iii) update and query sets for a mobile node must intersect. The update and query sets correspond to quorums. In the past, quorums have been used in distributed computing for mutual exclusion: in order to acess a shared resource in an exclusive mode the requesting node needs to receive permission from a quorum of nodes, and each node can grant only one request at a time. As two quorums always intersect, for every pair of concurrent resource requests there will be at least one node that will receive both the requests. This node will act as a tie-breaker for the two requests, sending a grant to only one of them. Thus, mutual exclusion is guaranteed.

In this paper we use the same notion of quorums in a different context of location management.

Mobile nodes exhibit a spatial locality of reference: even though all nodes in the system can potentially communicate with a given node, bulk of the references for the given node originate from only a subset of nodes (referred to as the *working set* in [25]). The nodes in the working set may be clustered in different parts of the network. So, to reduce query costs, it is advisable to have location servers for the MN in the vicinity of such clusters. Using only the MN's identity to determine its location servers fails to exploit the locality of reference characteristics.

Determining the location servers of an MN based solely on MN's location will lead to uneven distribution of responsibility. Often a significant fraction of MNs are concentrated in a very small area, while there is a very low density of MNs in the rest of the network. For example, most of the MNs may be situated on the highways and other major streets of a city during the morning and evening rush-hour traffic, and most of the MNs may be concentrated in the business districts of the city during rest of the day. In such situations, the directory servers [4] and reporting centers [5] in the high density RAs will be overburdened, while the directory servers and reporting centers in other RAs will be comparatively lightly loaded. Also, the area of MN concentration changes with time of day. So, equipping each area with enough servers to handle the peak load will mean that for most of the day these servers will be underutilized.

Hence, it is desirable that the location servers storing the location information of an MN be a function (h) of the MN as well as the cell in which the MN is present, i.e., $h : \mathrm{MSS} \times \mathrm{MN} \rightarrow S_{\mathrm{LS}}$. Here MSS denotes the mobile service station of MN's cell, and S_{LS} denotes a set of location servers. An MN's location servers will change as it moves from one *registration area* (RA) to another. Also, different MNs in the same cell may have different sets of location servers. This will result in better load distribution.

Nodes that wish to locate an MN should be able to access at least some of the location servers of the MN quickly, and in an inexpensive fashion. Function $h(\cdot)$, described above, can be employed to determine the set of location servers that should be queried to locate an MN if the information is not available at the location server of the local zone. The set $h(\mathrm{MSS}, \mathrm{MN})$ can represent the location servers that a node, in the cell represented by MSS, should query when it wishes to locate a mobile node MN. Thus, function $h(\cdot)$ determines the *update set* when an MN moves, and the *query set* for the location of the MN. The query and update sets (*quorums*) for every (MSS, MN) pair intersect. So, the latest location information of an MN can be accessed in a single round of message exchange.

The current HLR-based location management scheme can be interpreted as a special case of the proposed scheme where the first parameter of $h(\cdot)$, namely MSS, is inconsequential. In such a situation, $h(\mathrm{MSS}, \mathrm{MN}) \rightarrow$ HLR of MN. So, it is possible to implement the proposed solution in existing systems.

In first and second generation cellular systems the simplifying hypothesis of uniform user mobility was frequently assumed, due to the fact that the user community mainly consisted of business users with a very high mobility [12]. For systems such as Universal Mobile Telecommunication System (UMTS) [7], this hypothesis can no longer be considered valid, due to the much greater user penetration, comparable to that for fixed networks; in this case the presence of high shares of users with limited mobility can be envisaged [12]. Studies carried out on current mobile systems show that on average calling users do not address their calls uniformly over the territory, but tend to call users registered close to their own home node [8]. In future systems this fact will be stressed even further due to user behavior which will be more and more similar to that in current fixed networks [12].

So, it does not make sense to query the entire query set to find the target MN each time a call is placed. The update set for an MN consists of the location server for the MN's current zone and the set of servers returned by $h(\cdot)$. During location queries, first the querying node's zonal location server is probed. If the queried MN is in the same zone, location information is returned by the zonal server. Due to the locality of reference, a significant number of queries may be satisfied by the zonal server. Otherwise, the set of servers determined by function $h(\cdot)$ is queried.

4.2. Quorum selection

If each location server is contained in roughly the same number of quorums, then load balancing among location servers can be achieved by ensuring that the function $h(\cdot)$ maps on to each quorum with equal likelihood: a uniform hashing function. Here load reflects the number of updates and queries.

However, a uniform hashing function that ensures load balancing at a particular time cannot guarantee load balancing among location servers at other times. A group of new MNs may join the system. These new MNs and the nodes querying their location may be such that the function $h(\cdot)$ maps their location operations more frequently onto some quorums than others. In such a situation some of the location servers get overloaded while others are not so heavily loaded.

4.3. Load balancing with dynamic hashing

A solution to the problem mentioned above is to employ a family of universal hash functions [9]. Periodically, the system can switch from one hash function in the family to the other. This ensures that even if a particular hash function does not provide good load balancing under certain circumstances, over the long run load balancing among location servers is achieved. However, there are certain problems:

- Switching between hash functions may result in an (MSS, MN) pair to be hashed to a quorum that is different from the quorum selected by the previous hash function. This may require reorganizing information about all the MNs at all the location servers: an expensive solution.
- Long term load balancing may be acceptable from the point of view of the location servers. However, during the switch, location updates and queries will have to be suspended. From the user's perspective, temporary outages in location service may not be acceptable.

A better solution is to employ dynamic hashing. We define $h(\text{MSS}, \text{MN})$ to be a hash function whose range of values can expand or contract, depending on the load in the system. Each value in the range of function $h(\cdot)$ corresponds to a quorum. It is to be noted that in the proposed system model, besides a location server for each zone, there are also other general purpose servers. Under moderate to low load situations quorums consisting of only the zonal servers are in use. When overall load increases on the zonal location servers, the general purpose servers are also deployed, creating new quorums. Hence, the load on individual location servers declines. Conversely, when the frequency of location updates and queries goes down the general purpose servers involved in location management can be released leading to fewer and/or smaller quorums. These released servers can then be used for other services. The size and composition of quorums will be described in section 7.

5. Location update and query

5.1. Determination of location servers

1. Given an *MSS_id*, denoting the cell in which the mobile node is present, and an *MN_id* for that mobile node, we employ double hashing [9], as follows:

$$\begin{aligned} &h(\mathit{MSS\_id}, \mathit{MN\_id}) \\ &\quad = \big(h'(\mathit{MSS\_id}) + \mathit{MN\_id} \times h''(\mathit{MSS\_id})\big) \bmod m, \end{aligned}$$

   where $[0, m-1]$ is the range of hash functions $h(\cdot)$, $h'(\cdot)$, and $h''(\cdot)$. Functions $h'(\cdot)$ and $h''(\cdot)$ are uniformly distributed over the range $[0, m-1]$, and $h''(\mathit{MSS\_id})$ is relatively prime to m. Therefore, $h(\mathit{MSS\_id}, \mathit{MN\_id})$ will be uniformly distributed over the range $[0, m-1]$ [9].

2. Corresponding to each $i = h(\mathit{MSS\_id}, \mathit{MN\_id})$, there is a set S_i of location servers such that:
   (a) $S_i \not\subset S_j$, for $0 \leqslant i, j \leqslant m-1, i \neq j$.
   (b) $S_i \cap S_j \neq \emptyset$, for $0 \leqslant i, j \leqslant m-1$.
   (c) $|S_0| = \cdots = |S_{m-1}| = K$.
   (d) Any location server is contained in K S_i's, $0 \leqslant i \leqslant m-1$.

   Properties (c) and (d) represent the *equal effort* and *equal responsibility* properties, respectively. Together they represent the *symmetry* property.[3]
   So, if an MN with identity *MN_id*, in the cell corresponding to *MSS_id*, updates its location information, the update is done at the server for the local zone and all location servers in the set S_j such that $j =$

[3] Strictly enforcing the symmetry property reduces flexibility in constituting quorums. Hence, we shall later relax the symmetry property, in section 7, to construct quorums of different sizes, some of which have cardinality $O(\lg N)$, where N is the total number of zonal and general purpose location servers. Having such small quorums will lead to low overheads for location update and query operations.

$h(MSS\_id, MN\_id)$. S_j is referred to as the *update* set. As j is uniformly distributed over the range $[0, m-1]$, if the quorums are symmetric, the responsibility for location management is uniformly distributed among the location servers.

To perform location updates and queries the following types of messages are used: QUERY, RESPONSE, ADD, DELETE, and REPLACE. Their meaning will be clear in the following description.

5.2. Updating mobile node location

Let *old_MSS* be the MSS of the cell where the MN was located when the MN initiated the previous location update. Let *new_MSS* be the MSS of the cell in which the MN is present when it is about to initiate the latest location update. Before we describe the operations let us define some terms:

***purge_set*:** Set of location servers containing outdated location information of the MN. The outdated location information needs to be deleted from this set of servers.

***inform_set*:** Set of location servers that should store the new location information for the MN.

The purge and inform sets correspond to the old and new update sets of an MN, respectively. Note that if a location server S belongs to both the purge and the inform sets of an MN, then the outdated location information should be *replaced* with the new location information of the MN at S.

The following operations are performed to update the location information of the MN when it moves from the previous registration area to the new registration area:

1. *purge_set* $\leftarrow$ zonal server for old location; *inform_set* $\leftarrow$ zonal server for new location.
2. $j \leftarrow h(old\_MSS, MN\_id)$; $purge\_set \leftarrow purge\_set \cup S_j$;
3. $j \leftarrow h(new\_MSS, MN\_id)$; $inform\_set \leftarrow inform\_set \cup S_j$;
4. *new_MSS* sends DELETE message, containing *MN_id*, to all location servers belonging to the set difference represented by $purge\_set - inform\_set$.
5. *new_MSS* sends ADD message, timestamped with MN's local clock value and containing *MN_id* and *new_MSS*, to all location servers in the set difference represented by $inform\_set - purge\_set$.
6. *new_MSS* sends REPLACE message, timestamped with MN's local clock value and containing *MN_id* and *new_MSS*, to all location servers in $inform\_set \cap purge\_set$.

When a location server receives a DELETE message it deletes location information about the *MN_id* carried in the message. On receiving an ADD message, a location server adds an entry for *MN_id* indicating *new_MSS* as its location at the time indicated by the timestamp. On receiving a REPLACE message for an MN, the *old_MSS* and old timestamp value are replaced by the *new_MSS* and new timestamp value in the MN's location entry, as shown in figure 1.

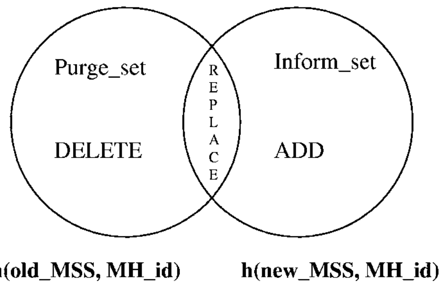

Figure 1. Location severs receiving ADD, DELETE and REPLACE messages.

5.3. Locating a mobile node

When a mobile service station with identity *MSS_id*, or a node inside the cell corresponding to this MSS wishes to locate an MN whose identity is *MN_id*, following actions are taken by the MSS:

1. Probe own zonal location server for *MN_id*'s location.
2. Exit if location information returned by zonal server. Otherwise, execute the following steps:
3. $j \leftarrow h(MSS\_id, MN\_id)$; $query\_set \leftarrow S_j$.
4. Send QUERY to all location servers $\in query\_set$ to locate MN with identity *MN_id*.
5. If a queried server contains location information about *MN_id*, it sends this information in its RESPONSE along with the associated timestamp. Otherwise, the servers sends a NULL response.
6. On receiving RESPONSE(s) containing location information, select those with the latest timestamp and extract the location information contained in them.[4]

As $S_i \cap S_j \neq \emptyset$ for $0 \leqslant i, j \leqslant m-1$, the *query* and the *update* sets for every pair of tuples (MSS_1, *MN_id*) and (MSS_2, *MN_id*), respectively, are bound to intersect. Therefore, every location query will return location information about the queried MN's RA.

5.4. Accuracy of location operations

Accuracy denotes the likelihood of a query for an MN returning the latest location information about that MN, stored anywhere in the system. Once location update operations (ADD, DELETE, REPLACE) have been performed at all the servers that belong to the *inform_set* and *purge_set*, only the latest location information is available at a subset of loca-

[4] A conservative termination strategy would be to wait for all RESPONSEs to arrive, whether NULL or non-NULL, and then determine the location. However, a faster approach would be to extract the location information from the first non-NULL RESPONSE. In section 5.4 we describe why this approach returns accurate information with very high probability.

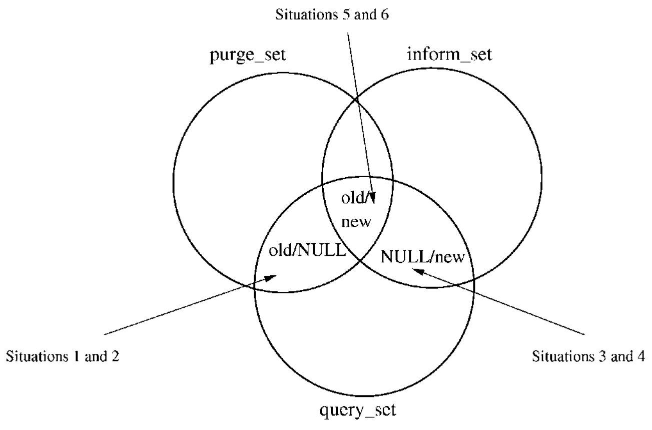

Figure 2. Impact of temporally overlapping location updates and queries.

tion servers (*inform_set* of latest update). Hence, all RESPONSEs to queries will either be NULL or return the latest location information, with at least one RESPONSE being non-NULL. So, queries return accurate location information.

However, let us consider the situation where a query for an MN's location is multicast to a quorum of servers, and that MN's location is being updated at the same time. Refer to figure 2. With regard to the location servers that belong to the *inform_set* and *purge_set*, there are six possible situations for the arrival of a location query message:

1. Location query reaches a location server belonging to (*purge_set* − *inform_set*) ∩ *query_set* before information is DELETED during location update: the location server sends outdated location information.
2. Query reaches location server belonging to (*purge_set* − *inform_set*) ∩ *query_set* after old location information is DELETED: location server sends a NULL reply.
3. Query reaches location server belonging to (*inform_set* − *purge_set*) ∩ *query_set* before location information is ADDED there: location server sends a NULL reply.
4. Query reaches location server belonging to (*inform_set* − *purge_set*) ∩ *query_set* after location information is ADDED there: location server sends latest location information in the reply.
5. Query reaches location server belonging to (*inform_set* ∩ *purge_set*) ∩ *query_set* before location information is REPLACED there: location server sends outdated location information in the reply.
6. Query reaches location server belonging to (*inform_set* ∩ *purge_set*) ∩ *query_set* after location information is REPLACED there: location server sends latest location information in the reply.

All other queried location servers, i.e., those belonging to *query_set* − (*purge_set* ∪ *inform_set*) return NULL replies.

Thus, for each of the three mutually disjoint sets: (i) (*purge_set* − *inform_set*) ∩ *query_set*, (ii) (*inform_set* − *purge_set*) ∩ *query_set*, and (iii) (*purge_set* ∩ *inform_set* ∩ *query_set*) there are two possibilities of the relative arrival time of location update and location query messages, leading to eight different combinations. Moreover, the intersection of the three sets, namely *purge_set*, *inform_set*, and *query_set* may be empty. After all, the quorum system does not explicitly state anything about the intersection of three quorums.

If the intersection of the three sets is empty situations 5 and 6 are not possible, leaving only four possible combinations of situations. Then all non-NULL responses return old location information when the following combination of situations occur: $(1, 3)$, and no information is returned if situations $(2, 3)$ occur. For the other two combinations, namely $(1, 4)$ and $(2, 4)$ new information is returned.

If the intersection of the three sets is non-empty, all non-NULL responses contain old location information if the following combination of situations occurs: $(1, 3, 5)$ or $(2, 3, 5)$. Under all other combinations of situations at least one RESPONSE contains the information stored by the latest location update operation. Assuming that local clocks of all nodes increase monotonically, outdated location information will have an earlier timestamp, and the latest location information will have a later timestamp. Therefore, the location information with the latest timestamp, which also happens to be the latest location information, is selected.

If old location information is retrieved by a query, the situation can be handled by temporarily maintaining a forwarding pointer at the old location while location information is being updated and for a short duration after that. The issue of when to purge stale forwarding pointers has been discussed in [19]. If no information is retrieved by a query, and no server has failed, it is a definite indication of the fact that the query for an MN's location was initiated while the corresponding information was being updated. So, the query is retransmitted. It is expected that within a small number of retransmissions (usually no more than one), servers in the *inform_set* will have acquired the new location information which will be returned by the query.

In summary, a location query has a very small probability of failing to return the latest location information: only during the time interval that a location update message is in transit from a cell to a location server in the *inform_set*. These messages travel along the high bandwidth fixed wireline network, So, we can safely assume that: (i) message propagation time is small, and (ii) the time between successive location updates for a mobile node is much greater than a message's propagation time. Therefore, most of the time location queries return the latest location information.[5] Results of simulation experiments, presented later in section 9.2, confirm that the error rate is extremely low for MNs moving at pedestrian speed as well as for MNs moving at highway speeds.

[5] If location update conditions proposed in [6] were to be employed then the interval between successive location updates for an MN would depend on the triggering threshold set for the time-based, number of movements-based, or distance-based policies (depending on which one is employed) and the mobility pattern of the MN.

6. Load balancing by dynamic hashing

Sometimes, even if the hash function is properly selected and the quorums are carefully formed, there is a possibility that some quorums may get more heavily loaded than other quorums. This problem can be handled by adding new location servers to the system when load increases. The new location servers, drawn from the pool of general purpose servers in the system, can be integrated with the existing location servers using *tries*, a dynamic hashing technique, as described in [11]. Basically, if a quorum of location servers is heavily loaded, a new quorum is added to the system. Location management responsibilities of the old quorum are now shared between the old quorum and the newly added quorum. Also, when load decreases, some of the location servers can be released and location management responsibilities of two lightly loaded quorums can be reassigned to just one quorum. Quorum addition and deletion, and integration of new location servers into the system can be accomplished seamlessly. Also, location information of only a small subset of mobile nodes has to be redistributed among the location servers during this process.

6.1. Dynamic quorum system approach

We assume that there exists a means to determine which quorums are heavily loaded (please refer to [24] for a discussion). Dynamic hashing (as described in [11]) is employed to relieve the load on heavily loaded quorums. The hash function $h(\cdot)$, stated in section 5.1, can be defined in a generic form as

$$h(a,b) = \big(h'(a) + b \times h''(a)\big) \bmod m.$$

Let m be a power of 2. Then we can define an entire series of hash functions, H, of the form:

$$h_i(a,b) = \big(h'(a) + b \times h''(a)\big) \bmod 2^i$$

for $0 \leqslant i \leqslant \mathit{limit}$, for some sufficiently large constant *limit*. Therefore,

$$h_i(a,b) = h_{i-1}(a,b)$$

or

$$h_i(a,b) = h_{i-1}(a,b) + 2^{i-1}.$$

So, given *MSS_id* and *MN_id*, the hash function h_i can be applied to map every (*MSS_id*, *MN_id*) pair to an integer in the range 0 to $2^i - 1$. Each integer corresponds to a quorum of location servers. Thus, each quorum in the location management strategy is similar to a bucket for holding data items that are hashed to the same value.

Each value returned by the hash function is also associated with a non-negative integer variable referred as *local_depth*. When hash function h_i is in use, the maximum permissible value of *local_depth* is i. Consistent information about *local_depth* is maintained at all the MSSs.

Initially, let hash function h_i be employed to map (*MSS_id*, *MN_id*) pairs to values in the range 0 to $2^i - 1$. Each of these values, v, has its *local_depth* set to i, and there is a *one to one* mapping between a value and a quorum. Let us refer to the quorum of location servers corresponding to the value v as S_v. Let us assume that up to 2^{limit} distinct quorums of location servers can be formed.

Let the cumulative update and query rates (load) corresponding to S_v exceed a pre-specified upper threshold. This is similar to a bucket overflow in the context of hash function. In such a situation, the action taken depends on the *local_depth* of S_v. There are two distinct cases with respect to *local_depth*:

6.1.1. Case 1: local_depth equals i

Let $\mathit{local\_depth}(v) = i$ at the time when S_v is overloaded. Then, the location management scheme switches from hash function h_i to the next higher hash function h_{i+1}. As a result of the switch to a higher hash function, if $h_i(\mathit{MSS\_id}, \mathit{MN\_id}) = v$, then:

$$h_{i+1}(\mathit{MSS\_id}, \mathit{MN\_id}) = v,$$

or

$$h_{i+1}(\mathit{MSS\_id}, \mathit{MN\_id}) = v + 2^i.$$

Also, $\mathit{local\_depth}(v)$ and $\mathit{local\_depth}(v + 2^i)$ are set to $i + 1$. For all values $0 \leqslant w \neq v < 2^i$ the value of $\mathit{local\_depth}(w)$ remains unchanged. For all newly created values $2^i \leqslant w \neq v + 2^i < 2^{i+1}$ in the range of function h_{i+1}, $\mathit{local\_depth}(w) = \mathit{local\_depth}(w - 2^i)$.

Also, if value w was previously mapped to quorum S_w, then as a result of the split, hash values w and $w + 2^i$ are mapped to quorums $S_{w \bmod 2^{\mathit{local\_depth}(w)}}$ and $S_{(w+2^i) \bmod 2^{\mathit{local\_depth}(w)}}$, respectively.

Assertion 1. As a result of the hash function switch, the (*MSS_id*, *MN_id*) pairs that were previously mapped to the heavily loaded quorum S_v are now split between two distinct quorums, namely, S_v and S_{v+2^i}.

Assertion 2. After the hash function switch, all other (*MSS_id*, *MN_id*) pairs, that were not mapped to the heavily loaded quorum, are mapped to the same quorum as before.

Proofs of these assertions can be found in [24].

Inference. As a result of hash function switch from h_i to h_{i+1}, a quorum of location servers that was heavily loaded now has its load split between two quorums: the previous heavily loaded quorum and a new quorum. Location information of only a fraction of mobile nodes will have to be shifted: from the location servers belonging to a heavily loaded quorum to location servers belonging to the new quorum. Other mobile nodes and location servers are unaffected by the split.

6.1.2. Case 2: local_depth less than i

Let the hash function in use be h_i, quorum S_v be overloaded, and $\mathit{local\_depth}(v) = k < i$. By construction of the hashing scheme, this indicates that for all w: $0 \leqslant w < 2^i \wedge w = v + j \times 2^k$, where j is an integer, $local\_depth(w) = k$. This indicates that all (MSS_id, VMN_id) pairs that are hashed to values corresponding to w are mapped to the same quorum S_v.

In such a situation, switching up from hash function h_i to h_{i+1} is not required. Instead, the following steps are executed:

1. $local\_depth(w) = k + 1$, $\forall w$: $0 \leqslant w < 2^i \wedge w = v + j \times 2^k$, $j \in I$.
2. All (MSS_id, VMN_id) pairs such that $h_i(MSS\_id, VMN\_id) = w$, are now mapped to quorum $S_{w \bmod 2^{k+1}}$.

As a result, all (MSS_id, MN_id) pairs that were previously mapped to the heavily loaded quorum S_v are now distributed between quorum S_v and S_{v+2^k}. The mapping from other (MSS_id, MN_id) pairs to quorums remains unchanged.

6.2. Handling quorum switch

Let some (MSS_id, MN_id) pairs, previously mapped onto quorum S_v, now be mapped onto a new quorum S_{v+2^i}, where $i \geqslant 0$. Then location information has to be *reconfigured* for the mobile node whose identity is MN_id, and whose latest location update was made while it was present in the cell represented by MSS_id. Reconfiguration of location information is similar to location update described in section 5.2, except for one difference: *in location reconfiguration, the location information of mobile node(s) does not change, but the set of location servers storing this information may change.*

All location servers in the set $S_v - S_{v+2^i}$ are asked to DELETE location information about the mobile node MN. No action needs to be performed at location servers belonging to the set $S_v \cap S_{v+2^i}$. Location servers in the set $S_{v+2^i} - S_v$ are asked to ADD location information about MN. The timestamp associated with the MN's location information being added is the same as the timestamp associated with MN's location information in the location servers belonging to the set $S_v - S_{v+2^i}$.

6.2.1. Example

Let, initially, only four quorums be in use: $S_0 - S_3$. Also, let the local depth of each quorum as well as the global depth be equal to 2. Quorum S_2 gets heavily loaded. As local depth of S_2 is the same as the global depth, a switch is made from hash function h_2 to h_3. Now, the queries/updates previously meant for quorum S_2 are split between quorums S_2 and S_6. The location information of some MNs is reconfigured between S_2 and S_6 by sending appropriate ADD, DELETE, and NO CHANGE messages to servers in S_2 and S_6. Other MNs remain unaffected. The local depths of S_2 and S_6 are increased from 2 to 3, while the local depths of other quorums remain unchanged at 2.

Later, when quorum S_1 becomes heavily loaded, there is no need to switch from hash function h_3 to h_4. This is because the local depth of of S_1, which is 2, is lower than the global depth. A new quorum S_5 is added, location information of some MNs is reconfigured between S_1 and S_5, and both these quorums have their local depths increased from 2 to 3.

6.2.2. Quorum addition and deletion

It is to be noted that each time the load of a heavily loaded quorum is split between two quorums, we are in essence adding one new quorum to the quorum system. The addition of quorums can be handled in the following manner:

1. Given a set of location servers, construct a quorum system in advance consisting of 2^{limit} distinct quorums.
2. Initially, use only a subset of quorums from the quorum set for location management. Let us refer to the quorums in use as *active quorums* and the remaining quorums as *reserve quorums*.
3. As load increases, due to addition of new mobile nodes and/or increase in location query and update activity of existing mobile nodes, add quorums from the reserve quorum set to the active quorum set.

Thus, initially all the location servers may not be in use for location management. The set of active quorums may not fully cover the set of location servers. In low to moderate load situations, *active quorums* are composed of only the dedicated zonal location servers. With increasing load, as the number of required quorums increases, more and more general purpose servers (usable for location management, as described in section 2) are pressed into service. This seems to be contrary to the desirable properties of *equal responsibility* and *equal effort* on the part of *all* location servers. So, we restate the properties as follows:

- All the location servers *in use at any given point of time* share equal responsibility and expend equal effort in location management.
- If the load (effort) of a non-empty subset of active location servers exceeds a pre-specified threshold, additional location servers are activated. The responsibility for location management is now shared among a larger number of location servers. As a result, the load on each server is reduced.

Just as the set of active quorums expands with increasing load, the set shrinks with decreasing load. When the rate of location update and query operations declines, some quorums can leave the active set and join the reserve set.

7. Quorum construction

The performance of the location management scheme depends on the size of the *query* and *update* sets (quorums). First, we will briefly describe three simple quorum construction strategies. Quorums so generated are symmetric. We will then describe yet another quorum system that compromises symmetry for smaller quorums. The communication and storage overheads for the smaller quorums are comparable to those for hierarchical, tree-based, location management schemes. Also, the upper bound on location query time is reduced to O(1) round-trip message delays as compared to $O(\lg N)$ for tree-based schemes, where the total number of location servers in the network is N.

7.1. Grid-based scheme

Let $N = l^2$ for an integer l. An $l \times l$ grid is constructed. The grid points are numbered from 0 to $N-1$. A quorum consists of all the grid points on a row, and one grid point on all the other rows. Thus, the cardinality of each quorum S_i is equal to $2\sqrt{N} - 1$. If N is not a square of an integer, a degenerate grid can be constructed with the outermost row/column being reduced in size. In such a case, while constructing S_i, the partial row/column is complemented using grid points from another row/column. Therefore, the size of each *query* and *update* set is $O(\sqrt{N})$.

7.2. Reduced overhead grid-based scheme

The degree of replication of location information for every MN can be nearly halved by adopting the following strategy described in [21]:

1. Once again all the location servers are arranged in a square grid.
2. There are $2\sqrt{N}$ quorums in all. However, they are distinguished as $\sqrt{N}$ update quorums and $\sqrt{N}$ query quorums.
3. The range of the hash function $h(\cdot)$ is $[0, \sqrt{N} - 1]$.
4. $query_i$ is the ith query quorum, where $0 \leqslant i < \sqrt{N}$, and consists of all location servers in the ith row of the square grid.
5. $update_i$ is the ith update quorum, and consists of all location servers in the ith column.

For all i, j, such that $0 \leqslant i, j < \sqrt{N}$, $query_i$ and $update_j$ always intersect in exactly one location server.

Thus, compared to the simple grid-based scheme, the storage and communication overheads are reduced. However, in the simple grid-based scheme any two quorums have at least two location servers in common. So, failure of one of those common servers does not prevent location information from being accessed. Such redundancy is not present in the reduced overhead scheme.

As described in section 5.4, when the quorum systems generated by the strategies mentioned above are employed there is a very small possibility that a query may return a NULL response. In order to preclude such a possibility, the following simple extension to the grid based scheme can be used:

1			
2	3		
4	5		
6	7	8	
9	10	11	
12	13	14	
15	16	17	
18	19	20	21
22	23	24	25
26	27	28	29
30	31	32	33
34	35	36	37
38	39	40	41
42	43	44	45
46	47	48	49

Figure 3. A CWlog quorum system with $N = 49$.

7.2.1. Extended grid-based scheme

Let $N = m^3$ for an integer m. An $m \times m \times m$ grid is constructed with the grid points corresponding to location servers. A quorum can be formed by taking all the grid points that belong to three mutually perpendicular planes in the grid. So, the size of each quorum is $O(N^{2/3})$, and *any three quorums will have at least one location server in common.*

7.3. Crumbling walls: efficient quorum system

The grid based scheme and several other quorum systems generate fairly large sized quorums. Moreover, as stated in [23], they are asymptotic in nature, manifesting their optimality only when the value of N is very large. We propose to use a particular type of *crumbling walls* quorum system, called the CWlog quorum system described in [23]. Like the grid based schemes, the location servers are logically arranged in rows. However, row widths are not uniform. The number of servers in the ith row is $\lfloor \lg 2i \rfloor$.

A quorum is the union of all location servers in one full row and a single representative from every row below the full row. In the CWlog system several quorums of size as small as $\lg N$ can be formed. This is a significant improvement over the grid-based schemes where each quorum is of size $O(\sqrt{N})$. Let us consider a 49-server system as shown in figure 3. The smallest quorum of size 4 consists of servers 46–49. Four quorums of five servers each, sixteen quorums of six servers each, and sixty-four quorums of seven servers each can be formed. In comparison to the grid-based scheme described earlier, where all quorums will be of size 13, more than 280,000 quorums of size 12 or smaller can be formed. In the CWlog system with 49 servers, the largest quorum size is 15.

8. Performance analysis

8.1. Time for location operations

A message round consists of multicasting a message and receiving a response. For a location update at most three multicast messages (ADD, DELETE, and REPLACE) to three disjoint sets of location servers are needed. As there are no causal dependencies between these messages they can be sent concurrently. *Hence, updates take one round-trip message time.* This is comparable to the hierarchical tree-based schemes where location of an MN is updated at all location servers along the path from the MN (a leaf node) to the root node.

If a node makes a location query for an MN in the same zone as the node, or if the location server of the node's zone belongs to the *update_set* of the MN, a look-up at the location server of the zone will provide the information. Otherwise, location information about the MN can be obtained, with a very high probability, in a single round of message exchange with a quorum of location servers. If all RESPONSEs are NULL another round of query will be made. By the time this query reaches servers in the *inform_set*, location information would have been ADDed there. *So, the time to locate a mobile node is bounded by two round-trip message delays plus the time to perform a lookup at the zonal location server of the local zone: a constant.* This is considerably lower than the hierarchical schemes where the bound is equal to $O(\lg N)$ round-trip message delays, N being the number of location servers. The constant time bound on location queries, in terms of network round-trip message delay, regardless of the number of location servers, indicates that the proposed scheme is scalable.

As the locality of queries increases, more queries can be satisfied by performing a look-up at the local server, and expected time declines. Also, as the call to mobility ratio increases, the probability that a query conincides with location update for the queried MN decreases. This means there is a lower probability of retries, resulting in reduced query time.

8.2. Communication overheads

The message complexity of each location operation is $O(\sqrt{N})$ if the grid-based scheme is used. If the CWlog quorum system is used, the size of the multicast set (quorum size) can vary between $\lg N - \lg\lg N$ (if the last row is used as quorum) and $N/\lg N$. However, a large number of quorums, and therefore multicast sets, of size $O(\lg N)$ are available. So, most of the time communication overheads are of the order $O(\lg N)$.

Each message has a small size, consisting of at most two MSS identities, one MN identity, a quorum identity, and a flag indicating the nature of the message (query, ADD, REPLACE, etc.). Moreover, these messages are transmitted in the fixed wire network, which has a much higher bandwidth than the wireless MN–MSS links.

For the *reduced overhead grid-based scheme*, even though the size of each quorum is $\sqrt{N}$, in several instances the actual communication overhead is significantly less than $\sqrt{N}$ times the communication overhead of the HLR/VLR based scheme. Let us illustrate the fact with the help of a simple example. Under the proposed scheme, let the geographical position of location servers also be such as to form a $\sqrt{N} \times \sqrt{N}$ grid. Let the corresponding HLR/VLR system consist of N independent networks integrated together such that the N HLRs are also located in a similar grid pattern. Let the communication overhead to traverse each link of the grid be *one unit*.

In the HLR/VLR based scheme a query or an update message may originate at any point of the grid and may be destined for any arbitrary HLR. Therefore, the expected communication overhead incurred by a message is proportional to the average distance between two arbitrary grid points. Also, in the proposed scheme using the reduced overhead grid-based quorum system let an ADD message originate at an arbitrary grid point and be sent to an arbitrary column of location servers. Such a message can be propagated by first routing the message to the nearest grid point for the destination column. In a $\sqrt{N} \times \sqrt{N}$ grid, the expected communication overhead for such routing is $(\sqrt{N} + 1)/3$. Then, the message travels along at most $\sqrt{N} - 1$ links to reach all the location servers in that column. Therefore, the total cost is equal to $(4\sqrt{N} - 2)/3$. Conceptually, this is akin to a broadcast along a tree consisting of the node that is the origin of the message and a column/row of location servers. Table 1 compares these costs for the HLR/VLR based scheme and the proposed scheme for various values of N.

Indeed, the communication overhead of the proposed scheme is less than twice, and not $\sqrt{N}$, times as high as the HLR/VLR scheme. The extra communication overhead is offset by the increased speed of location updates and queries, load balancing, decentralization of control, and scalability provided by the proposed scheme, as shown later in section 9.

8.3. Storage overheads

Assuming that the number of mobile nodes in the network is M, each of the N participating location servers has to store location information for $O(M/\sqrt{N})$ mobile nodes for the simple grid-based approach. This is because the location information of each of the M mobile nodes is replicated $O(\sqrt{N})$ times in the location directory, and the entire directory is distributed uniformly over N location servers. Each of these $O(M/\sqrt{N})$ pieces of information contains the *MSS_id* for the cell in which the mobile node is present. Assuming there are m MSSs in the system, this piece of information requires $\lg(m)$ bits. Besides this location information, each location server has to store an extra $O(Q\sqrt{N})$ entries, each of size $\lg(m)$ bits, indicating memberships of

Table 1

N	Overhead: HLR/VLR scheme (C_h)	Overhead: Proposed scheme (C_p)	C_p/C_h
4	1	2	2.0
9	1.78	3.33	1.87
16	2.5	4.67	1.87
25	3.2	6.0	1.88
...	...	...	...
144	7.94	15.33	1.93

all the quorums. Q is the total number of quorums in the quorum system ($Q = 2^{limit}$), and as already mentioned $\sqrt{N}$ is the number of servers in each quorum.

If the CWlog quorum system is used, each location server may not store the same amount of location information because the quorums are not symmetric. So, storage overheads can be expressed in terms of the number of copies of location information maintained for an MN. This number varies between $\lg N - \lg\lg N$ and $N/\lg N$.

8.4. Fault-tolerance

All the quorum based protocols fall into the class of *replicated data management protocols*. These protocols provide limited fault-tolerance at low communication costs [1]. For example, in the grid based scheme, two quorums intersect at no fewer than two servers. The probability that a query succeeds equals the probability that at least one server common to the update and query quorums is alive. If all servers have failure probability of p, and server failures are independent, the probability of success of a query is $1 - p^2$.

In the HLR/VLR scheme we have to consider two possibilities of query failure:

(1) the queried node is in its home RA and the HLR has crashed,

(2) the node is in a foreign RA and either the HLR or the foreign RA's VLR or both have crashed.

Assuming that HLRs and VLRs have the same failure probability as the servers in the proposed scheme, the probability of a query succeeding in the first situation is $1 - p$. The probability of success in the second situation is $(1-p)^2$. Thus, the quorum based scheme has a greater tolerance to failures.

If server(s) crash, a high availability of update and query operations can be maintained by employing the notion of structured read and write quorums with respect to logical structures, as proposed by Agrawal and El Abbadi [1]. A detailed discussion is not within the scope of this paper.

9. Simulation experiments

9.1. Load balancing

Three experiments were performed to evaluate the load balancing capabilities of the proposed distributed location management scheme. Event-driven simulations were performed where each MN's call arrivals, call terminations, and location updates were treated as individual events. The scheme was also compared with the conventional location management algorithm, which is based on the use of HLRs.

9.1.1. Experiment #1: Control case

The first simulation serves as a control case. A hexagonal cellular system was simulated with 100 MNs traveling randomly in a 100 km × 100 km closed rectangular grid.[6] Each cell was equipped with an MSS, for a total of 5000 MSSs systemwide. The maximum velocity, $V_{\max}$, and the rate of the maximum angular change,[7] $\Theta_{\max}$, for all MNs were assumed to be 60 mph and 24 degrees, respectively. Velocity, direction, and position updates were performed every two seconds based on the following equations (for more details about this model see [22]):

$$v(t+\Delta t) = \min\bigl[\max\bigl(v(t)+\Delta v, 0\bigr), V_{\max}\bigr], \tag{1}$$
$$\Theta(t+\Delta t) = \Theta(t) + \Delta\Theta, \tag{2}$$
$$x(t+\Delta t) = x(t) + v(t)\cdot\cos\Theta(t), \tag{3}$$
$$y(t+\Delta t) = y(t) + v(t)\cdot\sin\Theta(t), \tag{4}$$

where $[x(t), y(t)]$ denotes the MN's position at time t, and $v(t)$ and $\Theta(t)$ are the velocity and direction of the MN at time t, respectively. Δv, the velocity change, was uniformly distributed in the interval $(-A_{\max}\cdot\Delta t, A_{\max}\cdot\Delta t)$, $A_{\max}$ being the maximum acceleration/deceleration of the MN (assumed to be 1.2 m/s^2). $\Delta\Theta$, the change in MN's direction, was uniformly distributed in the interval $(-\Theta_{\max}, \Theta_{\max})$.

The on-call (call duration) and off-call (idle times) intervals are assumed to be exponentially distributed with expected values of 3 minutes and 30 minutes, respectively.

Each mobile was assigned an identification number MID $= 0, 1, \ldots, 99$. Each base station at location (x, y) was also assigned a unique number given by $MSS\_id = 50x + \lceil (y+1)/2 \rceil$.

There were fifteen location servers (LS) in the system. We employed a variation of the grid based scheme for quorum construction due to its simplicity of implementation. Even though the communication overheads of the grid based scheme are greater than the communication overheads of the crumbling walls scheme, the former is good enough to demonstrate the load balancing features of the proposed location management scheme.

The fifteen location servers were arranged as the following six quorums of 5 servers each:[8]

$$\begin{aligned}
S_0 &= \{\mathrm{LS}_1, \mathrm{LS}_2, \mathrm{LS}_3, \mathrm{LS}_4, \mathrm{LS}_5\},\\
S_1 &= \{\mathrm{LS}_1, \mathrm{LS}_6, \mathrm{LS}_7, \mathrm{LS}_8, \mathrm{LS}_9\},\\
S_2 &= \{\mathrm{LS}_2, \mathrm{LS}_6, \mathrm{LS}_{10}, \mathrm{LS}_{11}, \mathrm{LS}_{12}\},\\
S_3 &= \{\mathrm{LS}_3, \mathrm{LS}_7, \mathrm{LS}_{10}, \mathrm{LS}_{13}, \mathrm{LS}_{14}\},\\
S_4 &= \{\mathrm{LS}_4, \mathrm{LS}_8, \mathrm{LS}_{11}, \mathrm{LS}_{13}, \mathrm{LS}_{15}\},\\
S_5 &= \{\mathrm{LS}_5, \mathrm{LS}_9, \mathrm{LS}_{12}, \mathrm{LS}_{14}, \mathrm{LS}_{15}\}.
\end{aligned}$$

[6] The meaning of a closed rectangular grid is that if a mobile exits from one side of the rectangle, it emerges from the opposite side with the same velocity and direction.

[7] All angles are measured relative to the positive x-axis.

[8] With the *reduced overhead grid based scheme* it is possible to arrange sixteen location servers into four *query* quorums and four *update* quorums of 4 servers each resulting in an immediate reduction of about 20% in the load on location servers, compared to the results presented in this section. Thus, the efficiency of the proposed scheme can be easily extended beyond the values presented in this section.

During position update, each MN was checked to see if it had crossed the cell boundary. If so, we obtained two sets of LSs, the quorums S_i and S_j where i and j were related to the mobile host identification number (*MN_id*), the previous base station number (*old_MSS*), and the new base station number (*new_MSS*) in the following manner:

$$i = (MN\_id + old\_MSS) \bmod 6,$$
$$j = (MN\_id + new\_MSS) \bmod 6.$$

Messages, consisting of DELETE, ADD, and UPDATE, were sent to appropriate servers in the set $S_i \cup S_j$. The *write counter* of each of these servers was increased by one.

When a call arrival event was served, the quorum S_k was selected as follows:

$$k = (MN\_id + MSS\_id) \bmod 6,$$

where *MSS_id* was randomly generated (to mimic call originating from a random location in the network), and *MN_id* was the mobile host being queried. The servers that belonged to S_k were then queried and the *read counter* of the queried location servers was incremented by one.

Following a 90-minute transient period after the start of the simulation run, data were collected for 300 hours of system operation and the update and query loads on each location server were computed. The results are shown in figure 4(a).

9.1.2. Experiment #2: Mixed call and mobility pattern

The second simulation was similar to the first, except that some MNs were assigned different calling and mobility patterns, as follows:

$$V_{\max} = 65 \text{ mph}, \quad \phi_{\max} = 2 \text{ deg/s},$$
$$\lambda = \frac{1}{3}\ \text{min}^{-1}, \quad \text{for } 0 \leqslant MN\_id \leqslant 5;$$
$$V_{\max} = 5 \text{ mph}, \quad \phi_{\max} = 10 \text{ deg/s},$$
$$\lambda = \frac{1}{120}\ \text{min}^{-1}, \quad \text{for } 6 \leqslant MN\_id \leqslant 99,$$

where λ is the average call arrival rate. The results of the experiment are shown in figure 4(b).

9.1.3. Experiment #3: Comparison with HLR scheme

In the third experiment, the conventional HLRs were used to track the location of MNs. There were 15 HLRs in the system. The mobility and call-pattern parameters of MNs were the same as in experiment #2. Each MN was assigned a fixed HLR, i.e., *MN_id* is assigned to HLR$_i$, where $i = \lceil (MN\_id + 1)/7 \rceil$.

When an MN crossed the boundary between adjacent cells, the corresponding HLR recorded the new location. When an MN was paged, the system performed one read operation at the corresponding HLR. At the end of the experiment, the write and read loads on each HLR were recorded. Figure 5 shows the results of the experiment.

9.1.4. Discussion

Both experiment #1 and experiment #2 have uniformly distributed write and read load on all location servers even though the system in experiment #2 contains some MNs with significantly different mobility and call-patterns. Under the same conditions, on the other hand, in experiment #3, HLR$_1$ which was assigned the highly active mobiles, has much higher load than the others. The conventional scheme binds one HLR to a fixed set of MNs. Therefore the load on the HLR depends entirely on the parameters of those MNs. The proposed algorithm uniformly distributes the updates and queries among all location servers, hence achieving load balance.

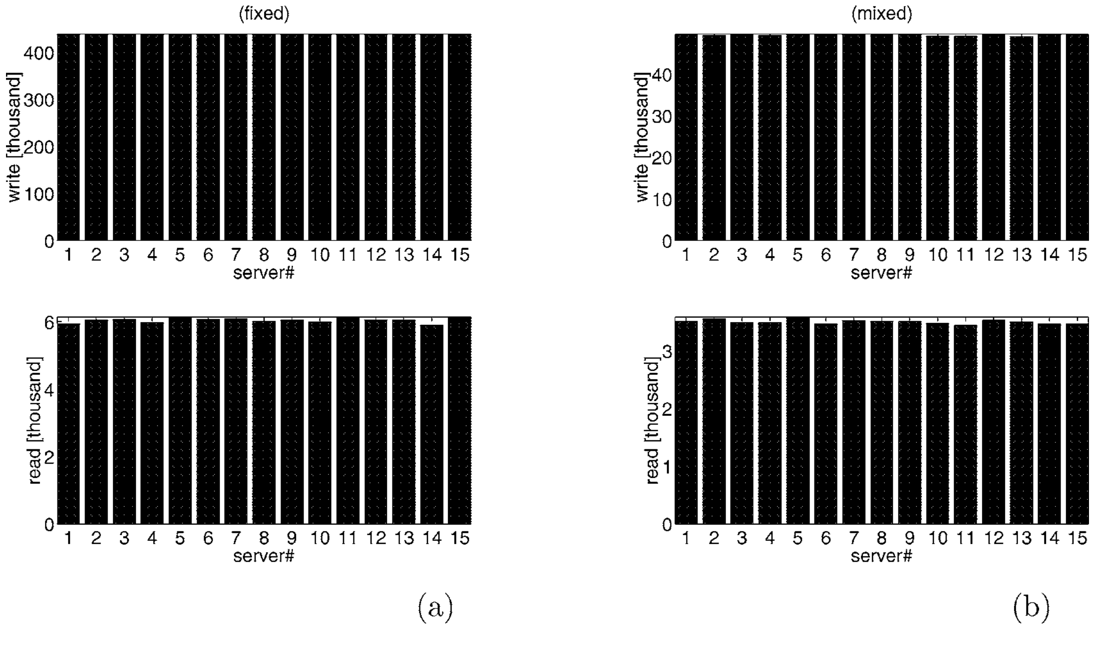

Figure 4. Load across servers with the load-balancing scheme and mixed call and mobility patterns.

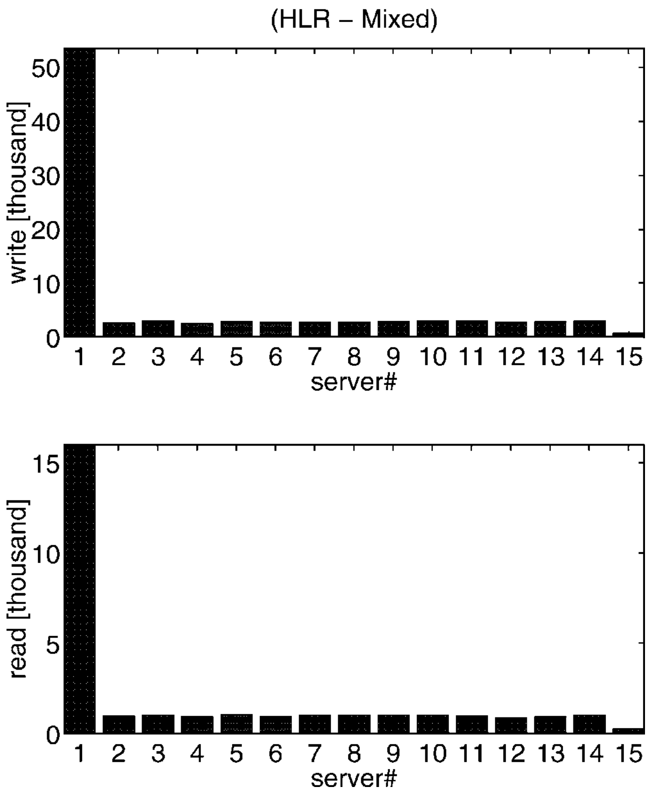

Figure 5. The histogram of load across servers: the traditional HLR mobility management with mixed call and mobility patterns.

The penalty for the achieved load balancing is that the average load on the servers is higher (figure 5). This is caused by the fact that, overall, there are more write and read operations, as each location update/search addresses a number of servers. However, the improvement is in the scheme's ability to eliminate overloading of some servers, while others remain underutilized.

Note that in the first two experiments locality of queries is not exploited as all queries are directly sent to the *query_set* without first querying the local zonal server. If locality were to be exploited, the read load on location servers would be further reduced, as described in section 8. Moreover, unlike the HLR/VLR scheme, the performance of the proposed scheme will improve as the system grows and the number of servers increases. This is because $2 \times \sqrt{N}$ as a fraction of N diminishes with increasing N.

9.2. Accuracy of location operations

As described in section 5.4, some delay occurs between the instant at which location update is initiated and the time the servers are updated. So, it is possible that a query will fail to produce any response or that the first response will point to an out-of-date location. We refer to these two cases as a read error, and investigated the relationship between the read error rate (number of read errors divided by the number of queries) and the time delay of the location server update.

We assumed the average time delay on all location servers to be the same. The read error rate was measured for delay

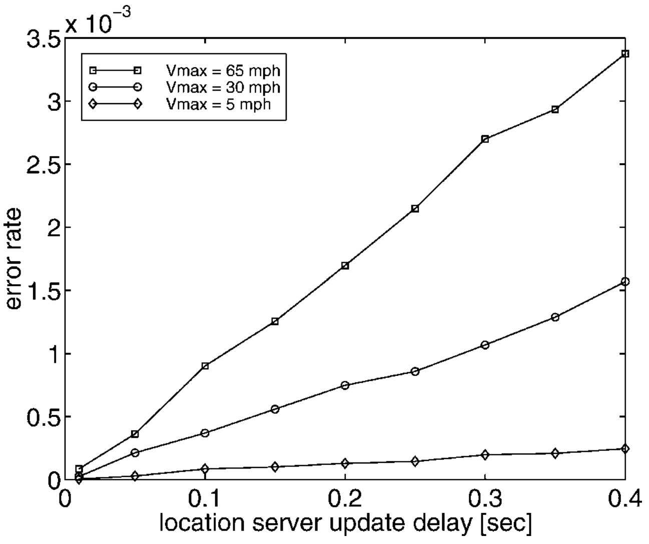

Figure 6. The rate of read failures as a function of update delay.

intervals from 50 to 400 ms in steps of 50 ms. Three simulations were performed using the mixed mobility and call patterns, as described earlier in section 9.1. To ensure good statistical averaging, 300 errors were collected in each run.

Figure 6 shows that the read failure probability is quite small for realistic values of parameters. The read error rate is an almost linearly increasing function with respect to the location server delay within 400 ms. Also, the error rate increases with MN's velocity, and the slope of the linear curve is approximately proportional to MN velocity.

9.3. Query delay

When a call is made to an MN, the system queries the location servers in a quorum. An alternative is to partition servers in a quorum into groups of some size and sequentially query the groups (by simultaneously querying all the servers in a group). We investigated the impact of group size on *read delay* (the number of group-read requests sent per call) and the *load per query* (the number of servers queried per call).

Quorums of size 12 were used, thus allowing group sizes of 1, 2, 3, 4, 6, and 12. In each run of the simulation using group size m, when a call arrived and a read quorum was determined, m servers were randomly chosen from the quorum. If an appropriate *MSS_id* was contained in one of these servers, the call was made successfully, otherwise, another group of m servers was randomly chosen from the remaining servers in the quorum, until an *MSS_id* was obtained. The total numbers of group-read and servers queried were recorded and divided by the total number of calls to determine the *read delay* and *load per query*.

Figure 7 shows the *read delay* and the *load per query* with a logarithmic x-axis and a linear y-axis. The *load per query* is the average number of server reads per query, while the *read delay* indicates the average number of groups polled

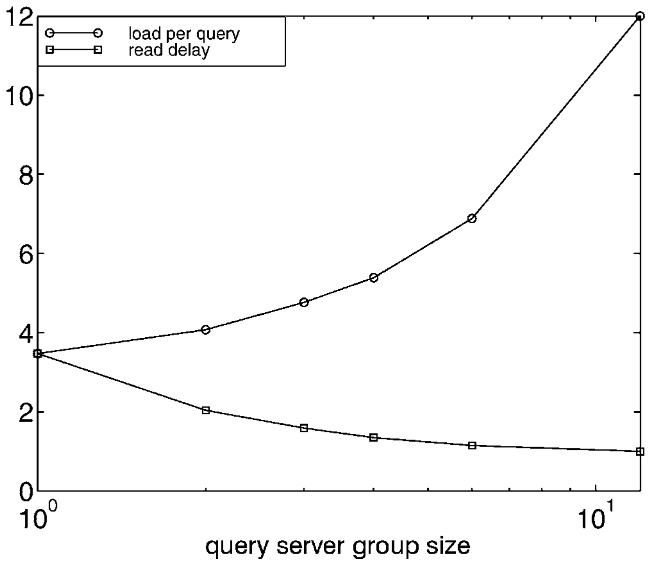

Figure 7. The load per query of and the read delay as a function of read group size.

per query.[9] The *read delay* is a decreasing function of group size, and that *load per query* is an increasing function of group size. Thus varying the group size allows trading the network bandwidth (*load per query*) with the speed of obtaining the search results (*read delay*).

10. Conclusion

We presented a distributed dynamic location management scheme. Dynamic hashing is used in conjunction with quorum systems to ensure fast update and retrieval of location information. The responsibility for location management is shared among all the servers in a fair manner. The proposed scheme does not require a home location register (HLR) to be associated with each mobile node. Location information is also *mobile* as it moves with the mobile node. Hence, the proposed scheme is especially suited for future systems employing *location independent numbering schemes*.

If a subset of location servers collectively gets overloaded with location management operations, their tasks are distributed among a larger set of location servers (quorum split) using dynamic hashing techniques. Also, if a set of location servers are very lightly loaded, some of them are released to perform other tasks and the remaining servers handle all location operations (quorum join). Thus, resources for location management, in the form of location servers, are dynamically allocated based on the demand. Location information of only a small subset of mobile nodes, affected by quorum split/join, has to be moved from one set of servers to another. All other mobile nodes remain unaffected. Thus, the system can gracefully adapt to changes in load.

Simulation results demonstrate that even if location query and updates are not uniformly distributed among all mobile nodes, each location server is equally loaded. This is in stark contrast to the HLR-based schemes where query and update loads are non-uniformly distributed among all the HLRs. The percentage of location queries that are incorrectly answered is negligibly small for the set of parameters studied. Also, varying the *query group size* results in a trade-off between the delay in obtaining search results and the control traffic in the network.

If the quorums are constructed so that servers in a quorum are uniformly distributed in the geographical coverage area then regardless of where the query originates, at least one location server of the queried quorum is close to it. So, there will be little variance in receiving a reply. On the other hand, with the HLR/VLR scheme there will be a greater variance in response time depending on the distance between the place where the query originates and the HLR.

A similar approach can also be used to solve the mobility management problem in other networking environments like mobile ad hoc networks. For example, a solution has been presented in [13]. Also, [18] presents quorum-based heuristics for information dissemination in a partitionable mobile ad hoc network.

[9] And is, thus, proportional to the time until the location information is found.

Acknowledgements

We thank the anonymous reviewers of the paper for their very helpful and constructive criticism. Their suggestions have enabled us to improve the quality of this work.

References

[1] D. Agrawal and A. El Abbadi, Resilient logical structures for efficient management of replicated data, in: *Proceedings of the 18th VLDB Conference*, Vancouver (1992) pp. 151–162.

[2] I.F. Akyildiz and J.S.M. Ho, On location management for personal communications networks, IEEE Communications Magazine (September 1996) 138–145.

[3] I.F. Akyildiz, J.S.M. Ho and Y.-B. Lin, Movement based location update and selective paging schemes, IEEE/ACM Transactions on Networking (August 1996).

[4] B. Awerbuch and D. Peleg, Online tracking of mobile users, Journal of the Association for Computing Machinery 42(5) (September 1995) 1021–1058.

[5] A. Bar-Noy and I. Kessler, Tracking mobile users in wireless communication networks, in: *Proceedings of IEEE INFOCOM* (1993) pp. 1232–1239.

[6] A. Bar-Noy, I. Kessler and M. Sidi, Mobile users: To update or not to update? in: *Proceedings of IEEE INFOCOM* (1994) pp. 570–576.

[7] S. Chia, The universal mobile telecommunications system, IEEE Communications Magazine 30(12) (December 1992) 54–62.

[8] G. Colombo, L. De Martino, C. Eynard and L. Gabrielli, Mobility control load in future personal communication networks, in: *Proceedings of the International Conference on Universal Personal Communication* (*ICUPC'93*), Ottawa, Canada (October 1993) pp. 113–117.

[9] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms* (MIT Press–McGraw-Hill, 1990).

[10] EIA/TIA IS-41 Rev. C, Cellular Radio Telecommunications Intersystem Operations, PN-2991 (November 1995).

[11] R.J. Enbody and H.C. Du, Dynamic hashing schemes, ACM Computing Surveys 20(2) (June 1988) 85–113.

[12] C. Eynard, M. Lenti, A. Lombardo, O. Marengo and S. Palazzo, A methodology for the performance evaluation of data query strategies in universal mobile telecommunication systems (UMTS), IEEE Journal on Selected Areas in Communications 13(5) (June 1995) 893–907.

[13] Z.J. Haas and B. Liang, Ad hoc location management using quorum systems, ACM/IEEE Transactions on Networking (April 1999).

[14] J.S.M. Ho and I.F. Akyildiz, Local anchor scheme for reducing location tracking costs in PCNS, in: *Proceedings of Mobicom* (ACM, 1995) pp. 181–193.

[15] R. Jain, Y.-B. Lin and S. Mohan, A caching strategy to reduce network impacts of PCS, IEEE Journal on Selected Areas in Communications 12(8) (October 1994) 1434–1444.

[16] R. Jain, Y.-B. Lin and S. Mohan, A forwarding strategy to reduce network impacts of PCS, in: *Proceedings of IEEE INFOCOM* (1995) pp. 481–489.

[17] R. Jain, S. Rajagopalan and L.F. Chang, Phone number portability for PCS systems with ATM backbones using distributed dynamic hashing, IEEE Journal on Selected Areas in Communications 15(1) (January 1997) 96–105.

[18] G. Karumanchi, S. Muralidharan and R. Prakash, Information dissemination in partitionable mobile ad hoc networks, in: *Proceedings of the IEEE Symposium on Reliable Distributed Systems* (*SRDS*), Lausanne, Switzerland (October 1999) pp. 4–13.

[19] P. Krishna, N.H. Vaidya and D.K. Pradhan, Static and adaptive location management in mobile wireless networks, Journal of Computer Communications 19(4) (March 1996).

[20] M. Mouly and M.B. Pautet, *The GSM System for Mobile Communications* (Palaiseau, France, 1992).

[21] S.J. Mullender and P.M.B. Vitányi, Distributed match-making, Algorithmica 3 (1988) 367–391.

[22] M.R. Pearlman and Z.J. Haas, Determining the optimal configuration for the zone routing protocol, IEEE Journal on Selected Areas in Communications 17(8), Special Issue on Ad Hoc Networks (August 1999).

[23] D. Peleg and A. Wool, Crumbling walls: A class of practical and efficient quorum systems, in: *Proceedings of the 14th ACM Symposium on Principles of Distributed Computing*, Ottawa (August 1995) pp. 120–129.

[24] R. Prakash, Z.J. Haas and M. Singhal, Load balanced location management for mobile systems using dynamic hashing and quorums, Technical report UTDCS-05-97, The University of Texas at Dallas (October 1997).

[25] S. Rajagopalan and B.R. Badrinath, An adaptive location management strategy for mobile IP, in: *Proceedings of ACM MOBICOM'95* (November 1995).

[26] C. Rose and R. Yates, Minimizing the average cost of paging under delay constraints, Wireless Networks 1(2) (July 1995) 211–219.

[27] N. Shivakumar and J. Widom, User profile replication for faster location lookup in mobile environments, in: *Proceedings of ACM MOBICOM'95* (November 1995) pp. 161–169.

[28] J.Z. Wang, A fully distributed location registration strategy for universal personal communication systems, IEEE Journal on Selected Areas in Communications 11 (August 1993) 850–860.

Ravi Prakash received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Delhi, in 1990, and the M.S. and Ph.D. degrees in computer and information science from the Ohio State University in 1991 and 1996, respectively. He joined the Computer Science Department at UT Dallas in July 1997 where he is an Assistant Professor. During 1996–1997 he was a Visiting Assistant Professor in the Computer Science Department at the University of Rochester. He was awarded the Presidential Fellowship by the Ohio State University for the year 1996. He is also the recipient of the best paper awards at several prestigious conferences. Ravi's areas of research are mobile computing, distributed computing, and operating systems. He has published his results in various journals and conferences.
E-mail: ravip@utdallas.edu

Zygmunt J. Haas (ACM member) received his B.Sc. in EE in 1979 and M.Sc. in EE in 1985. In 1988, he earned his Ph.D. from Stanford University and subsequently joined AT&T Bell Laboratories in the Network Research Department. There, he pursued research on wireless communications, mobility management, fast protocols, optical networks, and optical switching. From September 1994 till July 1995, Dr. Haas worked for the AT&T Wireless Center of Excellence, where he investigated various aspects of wireless and mobile networking, concentrating on TCP/IP networks. As of August 1995, he joined the faculty of the School of Electrical and Computer Engineering at Cornell University. Dr. Haas is an author of numerous technical papers and holds fourteen patents in the fields of high-speed networking, wireless networks, and optical switching. He has organized several workshops, delivered tutorials at major IEEE and ACM conferences, and serves as an editor of several journals. He has been a guest editor of three IEEE JSAC issues ("Gigabit Networks", "Mobile Computing Networks", and "Ad-Hoc Networks"). Dr. Haas is a Senior Member of IEEE, a member of ACM, and the Chair of the IEEE Technical Committee on Personal Communications. His interests include: mobile and wireless communication and networks, personal communication service, and high-speed communication and protocols.
E-mail: haas@ee.cornell.edu
WWW: http://www.ee.cornell.edu/~haas/wnl.html

Mukesh Singhal is a Full Professor of Computer and Information Science at the Ohio State University, Columbus. He received a Bachelor of Engineering degree in electronics and communication engineering with high distinction from University of Roorkee, Roorkee, India, in 1980 and a Ph.D. degree in computer science from University of Maryland, College Park, in May 1986. His current research interests include operating systems, database systems, distributed systems, performance modeling, mobile computing, and computer security. He has published over 140 refereed articles in these areas. He has co-authored two books titled *Advanced Concepts in Operating Systems* (McGraw-Hill, New York, 1994) and *Readings in Distributed Computing Systems* (IEEE Computer Society Press, 1993). He is a Fellow of IEEE. He is currently serving in the editorial board of EEE Transactions on Knowledge and Data Engineering and Computer Networks. He is also serving as a book series editor for a book series on distributed computing systems for the Oxford University Press. He served as the Program Chair of the 6th International Conference on Computer Communications and Networks, 1997, and of the 17th IEEE Symposium on Reliable Distributed Systems, 1998. He is currently serving as the Program Director of Operating Systems and Compilers program at National Science Foundation.
E-mail: singhal@cis.ohio-state.edu