An Efficient Fault-Tolerant Location Management Protocol for Personal Communication Networks

Karunaharan Ratnam, Sampath Rangarajan, and Anton T. Dahbura, *Fellow, IEEE*

***Abstract*—In this paper, we present an efficient fault-tolerant distributed location management protocol for personal communications service (PCS) networks. It achieves low connection-establishment delay, and under certain conditions, low overall cost compared to the current PCS location management protocol (i.e., IS-41). It also effectively avoids the shortcomings of IS-41, namely, centralized location management and triangular routing. Another feature of the protocol is its ability to recover from loss or corruption of the location information carried by the mobile host. For larger networks, this paper proposes two approaches to reduce the overhead of the distributed location management scheme. Further, we formally prove that our location management scheme maintains the correct location information of every subscriber in the system.**

I. Introduction

RECENT advances in communication technology, hardware, and software have created the opportunity for mobile terminals to receive many services that were, not long ago, only available to tethered terminals. Personal communication services (PCSs) are becoming a major force in providing these services to wireless users. To provide timely services to mobile users, a challenging task is to track the location of the mobile user efficiently so that the connection establishment delay is low. There have been several approaches proposed by various researchers for maintaining location information [2], [3], [8], [10], [11], [21], [1], [5], [9], [20]. In this paper, we present an efficient, fault-tolerant scheme for maintaining location information where the goal is to minimize the connection establishment delay.

In North America, the Telecommunications Industry Association's interim standard IS-41 [7], [6] is used for managing location information of roaming mobile users. IS-41 uses a two-tier approach for location tracking and maintenance (see Fig. 1). Every subscriber is registered with a location register (HLR) in the home network. The HLR maintains the subscribers current physical location. If the subscriber has roamed to another region, then he/she has to register with the visitor location register (VLR) that covers the new region. During the registration (and authentication), the VLR will contact the subscribers HLR, and the HLR will update its database to reflect the new location of the subscriber. The base station (BS) provides wireless connec-

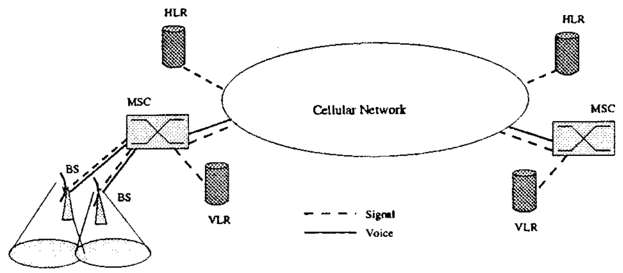

Fig. 1. Network reference model.

tion to the mobile user. The switching is performed at the mobile switching center (MSC).

The subscribers are assigned a ten-digit mobile identification number (MIN) by the PCS (or cellular) service provider. It is similar to a wireline telephone number. The MIN is programmed into the subscriber's mobile host before service is rendered to the customer. The MIN is the key field by which the subscriber's profile and location information are maintained. The MIN is transmitted by the mobile host over the air interface to identify itself to the network to obtain service.

In IS-41 [7], an incoming call is routed to the called subscriber as follows. The call initiated by dialing the subscribers's directory number DN (in most of the implementation, DN is the same as MIN, a ten-digit number). The first three digits of DN is the area code, and the next four digits indicate a specific MSC that serves the home subscribers. The dialed call is received by this MSC in the home system. This MSC is called the *originating MSC*. If the mobile host is currently in the originating MSCs coverage (i.e. the mobile host is not roaming), then the originating MSC pages the mobile host. When the mobile host responds (i.e., the subscriber accepts the call by pressing the right button), the originating MSC sets up circuits to terminate the call to the mobile host.

Fig. 2 shows typical call delivery to a roaming mobile host. As before, when a call to a mobile is dialed, the call is first routed to the originating MSC., which then sends a location request message to the HLR to find out the current location of the mobile. The HLR, in turn, sends a route request message to the VLR that is currently serving the mobile. The VLR then sends a route request message to the MSC that is currently serving the mobile. The serving MSC creates a temporary location directory number (TLDN) and returns it to the VLR. The TLDN is then passed back to the originating MSC through the HLR. The originating MSC then routes the call using this TLDN. When the serving MSC receives the call routed using the TLDN, it pages the mobile host. If the mobile responds, then the call is terminated to the mobile.

Manuscript received January 26, 1996; revised June 4, 1999. A preliminary version of this paper was presented at the 25th IEEE Symposium on Fault-Tolerant Computing, (FTCS-25).

K. Ratnam is with GTE laboratories, Waltham, MA 02451 USA (e-mail: karu@ece.neu.edu).

S. Rangarjan is with Bell Laboratories, Murray Hill, NJ 07974 USA (e-mail: sampath@research.bell-labs.com).

A. T. Dahbura is with AT&T Bell Laboratories, Hagertown, MD 21740 USA.

Publisher Item Identifier S 0018-9545(00)09178-7.

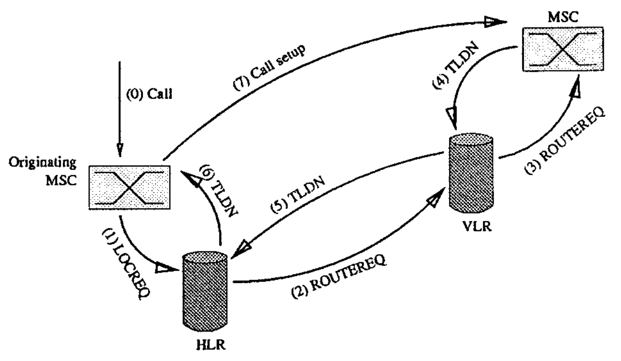

Fig. 2. Call delivery in IS-41 to an idle mobile in visited system

The HLR is a critical entity in the centralized IS-41 location management. There are many disadvantages in this approach. One is that any HLR system failure causes all mobiles registered with the HLR to be unreachable even though mobiles may be roaming and away from the HLR region. Thus, HLR is the single point of failure in the network. There is also another disadvantage, which is generally referred to as the *tromboning problem* or *triangular routing*. Consider the situation of a roaming mobile host. Subscriber MH-A's home MSC is in Seattle, WA, and MH-A is currently roaming in the Boston, MA, area. Assume that a user MH-B in Cambridge, which is a suburb of Boston, dials the number of MH-A to set up a call. This call setup involves two long-distance legs, one between Cambridge and Seattle, and the other between Seattle and Boston. The Seattle–Boston leg will be used twice, first to obtain TLDN and then for the voice trunk.

Several approaches have been proposed to alleviate these inefficiencies of IS-41. An improvement to IS-41 is proposed in [8], [10], and [11], where, depending on the call-to-mobility ratio,[1] the location information ($\mathrm{VLR_{id}}$) of a called mobile is cached at the calling MSC. That is, each time a call is made, the cached information is checked first. If the subscriber's current VLR information is cached, then that VLR is contacted directly, thus reducing the connection establishment delay. However, if the subscriber has moved out of that VLRs region, the HLR needs to be contacted after making an abortive connection request to the wrong VLR. In that case, the connection establishment delay will be longer than that of using the basic IS-41.

In IS-41, each time a mobile changes location and registers with a VLR, the HLR also need to be updated. As the VLR coverage shrinks to provide multimedia services, updating HLR for every location change imposes heavy traffic in the signaling network [15]. This traffic can be significantly reduced by not updating the HLR at each VLR crossings, rather, by maintaining a chain of forwarding pointers at each of the VLRs that the subscriber has visited [12], [13]. The HLR may then be updated after a threshold number of area crossings, or upon performing a search to contact the subscriber. A similar scheme, the *local anchoring scheme*, is proposed in [9] and [1]. In this scheme, a chain of forwarding pointers is not maintained, but a VLR closed to the current location of the subscriber is selected as a *local anchor*, which plays the role of HLR while the mobile is in the vicinity. All location changes are reported only to the local anchor. The HLR keeps a pointer to the local anchor. When a call arrives, the HLR queries the local anchor, which, in turn, queries the current VLR and obtains a TLDN for the call. In the above two approaches, if a subscriber is called very frequently from an area, each call requires a query to the callee's HLR and a traversal of the forwarding pointers. But, if the current location of the callee is stored by the frequent caller, the connection establishment time can be reduced.

[1]The ratio between the number of calls to a subscriber and the number of times the subscriber changes location.

In this paper, we propose a new approach that yields better connection establishment delay than previous approaches. In the fully distributed location management scheme we first present, the VLRs and HLRs are replaced by location registers (LRs). The LRs are distributed throughout the network. Similar to a VLR in the PCS architecture (Fig. 1), each LR serves a set of MSCs. The LRs can coexist with the MSC and serve only one MSC. There are no HLRs or VLRs in this scheme, and each LR maintains the location information of not only the mobiles that are local to it but also other mobiles in the network [16]. Thus, this is a protocol where the location information of the mobile hosts is fully replicated in all the LRs.

The LR functions as both a location registry for the local mobile hosts and the lookup directory for the location of other mobile hosts. For local mobile hosts, LR stores the current MSC serving the mobile. For mobile hosts that are not local, LR stores the ID of the LR where the mobile host currently resides. When a mobile registers with an LR, the new location information is disseminated to all other LRs in the network. This dissemination is carried out in parallel through the whole network so that new location information is very quickly updated at all the VLRs.

Of course, for reasonably large networks, neither fully replicating the location information nor disseminating location information (when a mobile changes location) through the whole network is practical. Hence, we discuss how the basic protocol can be tailored such that both the overhead due to replicating location information and the overhead due to dissemination are minimized.

In the next section, we describe our generic scheme based on fully distributed location management. We also formally prove the correctness of the scheme in the next section. In Section III, we show how the fully distributed scheme can be tailored for large networks. Conclusions are presented in Section IV.

II. Fully Distributed Location Management

A. Registration and Connection Establishment

Registration: The base stations periodically broadcast "beacon" signals to the mobile hosts (MHs) in its coverage area (a.k.a., cell). An MH needing connection can then send a request to the base station to be monitored by it. The request is in the form of a *monitor-me-join* (mmj) message (Fig. 3). Along with the mmj message, information about the identity of the MH and the location counter (LC) value (a sequence number that will be explained next) is also sent. Similarly, when an MH switches itself off and then wakes up, it sends a *monitor-me* (mm) message to the base station to monitor it. Thus, the base station can distinguish requests from mobile hosts that have just woken up from those that have moved into its coverage area.

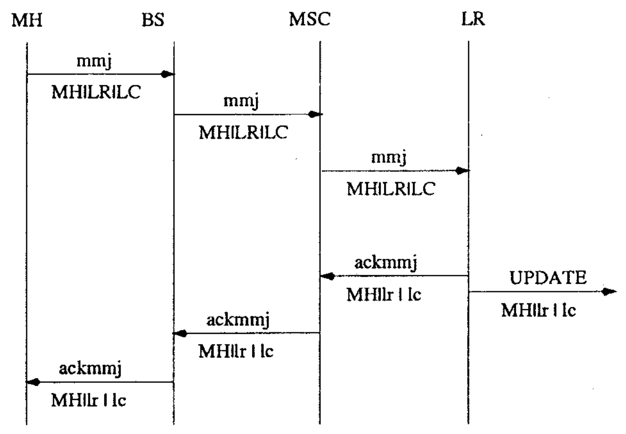

Fig. 3. Registration in fully distributed location management

In case the MH enters an area where the cells of more than one base station overlap, the MH may wish to change its monitor if it finds that another base station has a stronger beacon signal. If this monitor's beacon signal becomes weaker later, at that time it could send a request to another base station (if another base station with an overlapping cell is available) requesting to be monitored, and so on.

Upon receiving an *mmj* request from an MH, the base station will inform its MSC. This MSC, in turn, forwards the request to the LR associated with it. From its database, the LR retrieves the ID of the last LR that served this MH. If this LR is the one that last served the MH, then the LR will compare the LC value that arrives in the *mmj* message with the one in its database, assume the larger of the two to be the LC value, and send an *ackmmj* message along with the new LC value to the MH. The old LR value and the new LR value could be the same when the MH moves from one base station to another where both base stations are served by the same LR. If the LR that last served the MH is different, then the LR will increment the larger of the two LC values and send an *ackmmj* to the MH. From then on, whenever a message is sent from/to this MH, the message needs to be directed through this BS/MSC.

The MH should receive the *ackmmj* message within a certain period of time; if not, it times out and sends an *mmj* request to another overlapping base station if one exists. This procedure is repeated until an *ackmmj*is received. Once an *ackmmj* is received from a BS/MSC, the MH is then able to communicate with other hosts on the network through this BS/MSC. In the case of overlapping access zones, once the *ackmmj* message has been received and registration has been established through a new LR, the MH sends a *break* signal to the old LR to severe that connection. Together with the break signal, it also sends the identity of its new LR and an updated location counter value. The *break* signal is useful for immediately forwarding an incoming message to the MH if it has crossed an LR service area. When an LR sends an *ackmmj* back to the MH, it also updates its own location directory (LD) and disseminates information about the new location of the MH to the other LRs in the network.

It is to be noted here that an MH may cross several base-station cells before crossing an LR service area, as there are many base stations served by one LR. Only when an MH crosses over to another LR is the location information disseminated in the network.

Call Setup: For the protocol explained in this section, it is assumed that the location information of each mobile host is fully replicated in all the LRs. The MH need not be assigned to a particular home location register. Hence, unlike IS-41, a call to an MH can be directly sent to the serving LR instead of querying the HLR first. The serving LR contacts the serving MSC to assign a routing number (TLDN) to the call. However, due to the finite delay involved in completing a location information dissemination, when an originating MSC receives a call request to a nonlocal MH, there is a nonzero probability that the associated LR has old location information of the called MH. Because of this, location request for an MH could be received at an LR that is not serving this MH any more. In this case, the location request is forwarded to the LR that is currently serving the MH as per the location information available at the LR that received the location request. This way, through a chain of forwarding steps, the location request eventually reaches the LR that is currently serving the MH. It is to be noted here that, since many base stations are served by an LR, chances of a mobile host crossing over to another LR region while its earlier location update still propagating in the network is negligibly small.

B. Information Dissemination and Location Counter Corruption

Each MH has a location counter associated with it. The LC acts as a logical time stamp. Whenever an MH requests registration, the LR increments the corresponding LC value and sends it back to the MH. The LR then updates its own location directory and disseminates information about the new location of the MH together with the new LC value to the neighboring LRs. The LRs that receive this information determine if it is new (*newinfo*), old (*oldinfo*), or the same information (*sameinfo*), depending on whether the LC value they received is larger, smaller, or equal to the LC value they have locally stored for this MH. The LR propagates this new information to all its neighbors and updates its local LD. If the information is old, it (the LC value for this MH and its location) is updated and sent back to (only) the sender. If the information is the same, but the LR ID in the update message is different from the LR ID stored in the LD, then the information in the update message is added to the LD. Thus, the location counter serves to distinguish among new, old, and same information and thereby limits redundant messages that may be propagated on the network.

Given this functionality for the location counter, it is important to ensure that the location counter values contained locally at an LR and those carried in the information messages always monotonically increase. This property is required to enable more recent information to be associated with a larger LC value than that associated with older information. Informally, we would like to guarantee that newer information does not get overwritten by older information. When MHs move from one LR service area to another, they notify the new LR of their LC value. If this notified LC value is correct, then the above property can be shown to hold. Our protocol is tailored to recover from situations where this notified location counter value is incorrect. An LC value can become incorrect for two reasons.

1) A hardware failure can cause the MH to send an incorrect LC value. That is, the location counter value has been corrupted.
2) An MH sends an *mmj* message, and before receiving an *ackmmj,* it moves out of the cell and crosses over to a new LR region. In this case, if the location update information from the old LR arrived at the new LR before the new *mmj* message from the MH arrived, the LC value sent by the MH will be incorrect as far as the new MSC is concerned, even though there is no hardware failure on the part of MH.

Our protocol treats the latter case as if the LC value has been corrupted, because a protocol that can tolerate LC corruption can also tolerate the above mentioned *movement dependent* failure.

In addition to this, an MH could switch off and then reappear in another part of the network. As discussed earlier, it is possible that when an MH turns off power, its memory is not retained for an extended period. Hence, when this MH wakes up and wants to send an *mm* message, it may not have the ability to send its correct location counter value. As we will show later, this case is equivalent to the case where the location counter value is corrupted. That is, the scheme also handles the case where an MH turns off power, loses its memory, and then reappear elsewhere in the network.

Our scheme tolerates location counter corruption by making the incorrect location counter value converge to a consistent location counter value and stabilize itself. This is accomplished as follows. Assume that an MH moves from LR_i service area to LR_j service area. The location directory at LR_j will contain one of the following about this MH.

1) LR_j retains old information that this MH is being served by some LR_h. This could happen if LR_j has not yet received information that the MH moved into LR_i's coverage area.
2) LR_j retains the most recent information that it is being served by LR_i.

Consider the situation where the LC value carried by the MH gets corrupted and becomes incorrect. The MH sends this incorrect LC value (as part of the *mmj* message) to LR_j. In that case, if LR_j updates its location directory and disseminates this information, it is possible that this information is deemed *old* information by other LRs on the network and ignored. It is also possible that some other LR receives this as *same* information in case the LC value received is the same as the LC value stored for this MH in their local LD. Even though the location information is different, because the LC values are same, this information would be ignored unless this is perceived as new information. We take an optimistic approach and let LRs disseminate this information. After LR_j disseminates the new location information of the MH, the following three cases can happen.

1) The disseminated LC value is larger than the LC value that all other LRs have stored in their LD for this MH.
2) The disseminated LC value is smaller than the LC value that at least one LR has stored in its LD for this MH.
3) The disseminated LC value ties the largest LC value stored in all other LRs.

If the disseminated LC value is larger than the LC value that any other LR has stored for this MH, then this information will be correctly perceived as new information and all LDs will be correctly updated. If the disseminated LC value is smaller than any other LC value on the network, eventually this larger LC will reach LR_j, and at this time LR_j resolves the inconsistency by first checking if the MH is still in its service area (by sending a *check* message through the current base station). If the MH is still served by LR_j, it will send an *ackcheck* message back, and LR_j increments the LC value and disseminates this again on the network. If the MH is not in its service area anymore, LR_j does not have to do anything, as the inconsistency will be resolved by the LR who is currently serving the MH. In case the LC value ties the largest of the LC values that is stored in some LR, this situation is made known to LR_j so that it can resolve this inconsistency as in the previous case.

A similar situation arises when an MH wakes up after being switched off and sends a *monitor-me (mm)* message to some base station in the service area of LR_j. In this case, LR_j will disseminate *newinfo* about the location of this MH by incrementing the LC value that it had stored in its LD for this MH. If there are any inconsistencies, then it will be resolved exactly as described in the last paragraph. By the time the MH wakes up after being switched off, if the most recent LC value (the LC value just before it was switched off) has reached all the LRs on the network, then the incremented LC value that LR_j has disseminated will be larger than the LC value stored in the LD of any the LRs, and there will be no inconsistency.

C. Data Structures

The data structures required are as follows.

Location Directory: An LD is maintained at each LR in the network. It contains information about the location of each MH in the network. In the LD at (say) LR_i, an entry will look like $(\mathrm{MH}_{\mathrm{id}}|\mathrm{LR}_{\mathrm{id}}|\mathrm{LC})$. Let us refer to the entry for the jth mobile host in the LD of LR_i by $\mathrm{LD}_i[j]$ and to the individual fields in the entry by $\mathrm{LD}_i[j].field$. For example, $\mathrm{LD}_i[j]$.LC refers to the LC value for MH_j at LR_i. $\mathrm{MH}_{\mathrm{id}}$ refers to the ID. of the MH. $\mathrm{LR}_{\mathrm{id}}$ gives the location of the MH; that is, the ID of the LR that is serving this MH as per information that is available at LR_i. *LC* refers to the location counter value for the MH as per the information that is available at LR_i.

Messages Within the Signaling Network: Within the signaling network, information about the changing location of different MHs is disseminated as explained earlier. The data structure for the message that carries this information is quite similar to location directory. This message contains entries of the form $(\mathrm{MH}_{\mathrm{id}}|\mathrm{LR}_{\mathrm{id}}|\mathrm{LC}_c)$. To preserve bandwidth, a number of entries may be concatenated into an update message. The number of entries depends on the number of MHs about which new location information needs to be propagated.

Messages Within the Wireless Network: Within the wireless network, base stations and mobile hosts exchange messages. An *mmj* message from an MH to a base station has the structure $(\mathrm{MH}_{\mathrm{id}}|\mathrm{LC})$. The *mm* message contains only $(\mathrm{MH}_{\mathrm{id}})$. The *ackmmj, break, check,* and *ackcheck* messages are of the form $(\mathrm{MH}_{\mathrm{id}}|\mathrm{LR}_{\mathrm{id}}|\mathrm{LC})$.

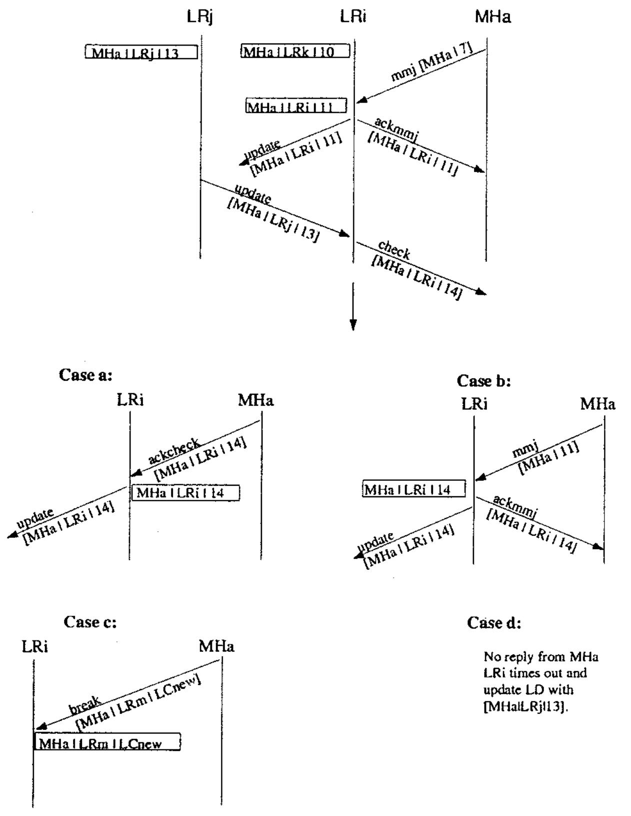

Fig. 4. Sequence of messages exchanged in scenario 1.

D. Correctness

The correctness of the protocol is provided later in this section. First, let us look at two example scenarios to see how the scheme works. These examples are explained using event-time diagrams to illustrate the progression of conflict resolution.

Scenario 1: The LC value that is carried by the mobile host could be corrupted due to any of the reasons explained earlier. The incorrect LC value could be larger, smaller, or same as the LC value stored by an LR that the MH wants to register. Fig. 4 shows the event-time diagram of the sequence of messages exchanged in the network when a mobile host A with an incorrect LC value tries to join an MSC in the service area of LR_i. Let the LC value carried by the mobile host be seven and the location directory entry for MH_a at LR_i be $(\mathrm{MH}_a|\mathrm{LR}_k|10)$. Upon receiving the *mmj* from MH_a, the MSC forwards the request to LR_i. Now, LR_i checks its LD to see whether MH_a is already registered with it through some other MSC. Let us assume that MH_a is not being served by LR_i. Now, LR_i has two LC values for MH_a: the value that arriveds with the *mmj* message and the one in its LD. LR_i will assume the larger of the two LC values (i.e., ten), increment it, and send an *ackmmj* message back to MH_a along with the incremented LC value. LR_i will also disseminate the new location information of MH_a along with the new LC value to all other LRs in the network. Once MH_a receives the *ackmmj* message, it will update its local LC value.

The LC value disseminated by LR_i may be smaller than what some other LRs (say, LR_j) have stored in their location directory. This will happen, for example, if the information disseminated by LR_j, to which MH_a had previously been registered, is yet to reach LR_i. Let the location update disseminated by LR_j be $(\mathrm{MH}_a|\mathrm{LR}_j|13)$. When LR_i receives the update from LR_j, it will increment that LC value and send a *check* message along with the incremented LC value (i.e., 14) to MH_a to verify that the MH is still connected to it. At this point, there are four possible scenarios that could occur.

1) MH_a *is still in the coverage of the same MSC and has not severed connection with it.* Since MH_a has neither moved out from the MSC's coverage nor severed connection with the associated LR, the MH will respond to the *check* message by sending an *ackcheck* message back to LR_i. Upon receiving the *ackcheck* message from MH_a, LR_i will update its location directory with the new LC value (i.e., 14) and disseminate this location information of MH_a on the wired network. This process will be repeated until one of the following events happens: 1) there is no LR with LC value for MH_a larger than what LR_i has stored in its LD or 2) MH_a has moved out of that MSC's coverage or established a connection through some other LR. In the first case, all LRs will eventually receive the location information of MH_a from LR_i. The latter case is analyzed next.
2) MH_a *has moved to the region of another MSC that is also in the service area of* LR_i. When MH_a joins the other MSC, the *mmj* message it has sent will be forwarded to LR_i. LR_i may or may not have received this *mmj* message before it times out on the *check* message. In either case, upon receiving the *mmj* request, LR_i will update the LC to 14 and sent an *ackmmj* along with the new LC value to MH_a. LR_i then disseminates the location information of MH_a along with the new LC to all other LRs.
3) MH_a *has moved to another service area and established registration with* LR_m, *and* MH_a *sent a break signal to* LR_i. The new LC value LC_{new}, which came along with the *break* signal, depends on what LC value LR_m had stored in its location directory for MH_a. If LC_{new} is larger than the LC value at LR_i, then LR_i updates its LD with $(\mathrm{MH}_a|\mathrm{LR}_m|\mathrm{LC}_{\mathrm{new}})$. If $\mathrm{LC}_{\mathrm{new}}$ is smaller, then LR_i eventually updates its LD with the LC value that arrives on the wired network from LR_m. In the latter case, it is LR_m's responsibility to resolve the correct LC value for MH_a and disseminate this information.
4) MH_a *has moved out and registered with some other LR without sending a break signal to* LR_i. In this case, once LR_i receives the LC value from LR_j, it will check to see whether MH_a is still in its region. Since MH_a has moved out, there will be no *ackcheck* message from MH_a. Hence, LR_i times out and does nothing. However, LR_i propagates the information it received on the wired network to other LRs to make sure that the new LR to which MH_a has moved is aware of the fact that there is an LR with a larger LC value for MH_a.

Scenario 2: Consider another scenario where MH_a registers with LR_i and its new LC value is updated to be 11. Now (see

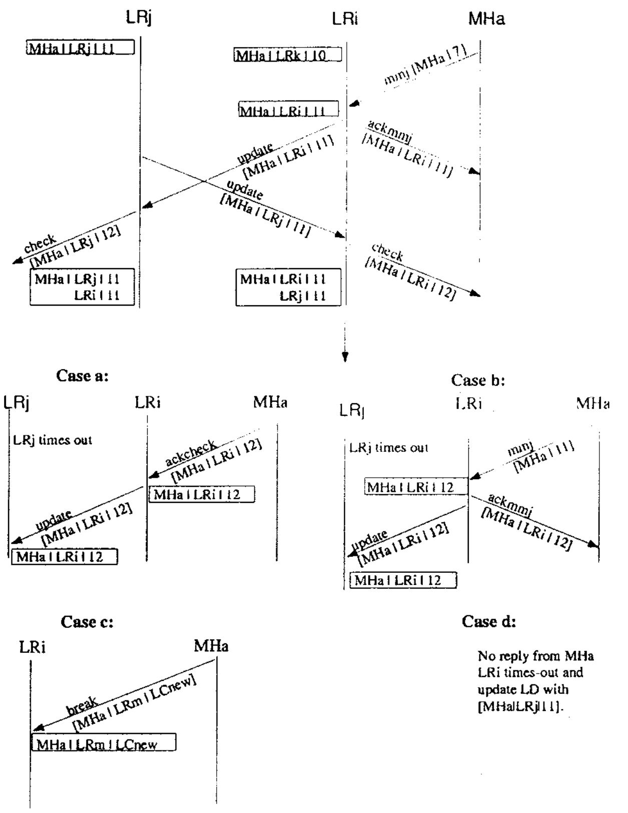

Fig. 5. Sequence of messages exchanged in scenario 2.

Fig. 5), LR_i receives the location information about MH_a disseminated by some other LR, which indicates that MH_a is connected to LR_j with LC value 11. Eventually, LR_j will also receive information originated by LR_i, which indicates that MH_a is connected to LR_i with LC value 11. Now, both LR_i and LR_j have the same LC value for MH_a, but different location information. Both LRs will increment this tied LC value and send a *check* message to MH_a along with the incremented LC value (i.e., 12). Each LR in the network (including LR_i and LR_j) that received this tied LD information for MH_a will update its LD for MH_a with $(MH_a|LR_i|11|LR_j|11)$ to indicate that there are two LRs with the same LC value that claim to have a connection with MH_a. This tie situation can only be resolved by the LR currently serving MH_a. Again, there are four possible scenarios that could occur.

1) MH_a *is still in the region of the same MSC that is connected to* LR_i *and* MH_a *has not severed the connection.* Since MH_a is still served by LR_i, it sends an *ackcheck* message to LR_i. Upon receiving the *ackcheck* message, LR_i will remove LR_j's name from the LD entry and disseminate the new location information on the wired network.

   Since MH_a is not in the service area of LR_j, it will time out on its *check* message. (Note: MH_a may have moved back in, but in that case, MH_a would send an *mmj* message first). Now, the location directory information for MH_a at LR_j will have both LR_i and LR_j in their LR_{id} field. Since LR_j cannot resolve the tie situation, it will propagate the location directory information that arrived on the wired network [i.e., $(MH_a|LR_i|11)$] to other LRs while keeping the entry $(MH_a|LR_i|11|LR_j|11)$ in its LD to indicate there are two LRs claiming responsibility for MH_a

   All other LRs except LR_i and LR_j will just propagate the LD update information since they are not currently serving to MH_a. If they receive the same information again (i.e., message saying that MH_a is in the region of LR_i or LR_j with the same LC value), they will ignore that information. Once LR_i resolves the tie, all other LRs (including LR_j) will remove LR_j from their LD entry for MH_a and update the LC value.
2) MH_a *has moved to the region of another MSC, which is also in the service area of* LR_i. In this case, LR_i will either time out on the *check* message before receiving MH_a's *mmj* request through the new MSC or LR_i will receive the *mmj* request before it timed out. In either case, upon receiving the *mmj* request, LR_i will update the LC value to 12 and send an *ackmmj* message to MH_a along with the new LC value. Also, LR_i will disseminate the location information of MH_a along with the new LC to all other LRs.
3) MH_a *has moved out of* LR_i*'s region and sent a break signal before leaving.* Since the LC value of MH_a before moving out of LR_i is equal to the LC value that LR_i has received on the wired network, the new LC value that came along with the *break* signal is strictly larger than what LR_i received. Hence, LR_i updates its LD entry for MH_a with $(MH_a|LR_m|\mathrm{LCnew})$. When LR_j receives the new information from LR_m, it will also update its location directory.
4) MH_a *moves out of the service area of* LR_i *without sending it a break signal.* In this case, both LR_i and LR_j time out on their *check* messages to MH_a. But, both of them propagate the LD information that came on the wired network. Once they receive the updated information from the new LR that MH_a moved into, LR_i and LR_j will update their LDs. All other LRs do likewise. The new LC value is strictly larger than what LR_i and LR_j had stored earlier in their LD entry for MH_a.

   Thus, if the LC value is incorrect due to any of the reasons discussed earlier, the protocol will eventually stabilize with the correct LC value, and all LRs will eventually have the correct location information for the MH.

Proof of Correctness: We need to show that, after the *last movement* has occurred in the network, each LR's location directory will *eventually* have the same and correct *location counter* value and the serving LR_{id} for all the active mobiles. Further, we also need to show that there is no *live-lock* in the system. That is, we need to show that after the *last movement, eventually* no location update message is exchanged in the network except for the periodic *beacon* signals that are sent over the wireless channel.

Here we do not make a distinction between the active and standby modes of operations of a mobile host. We need to show that these properties of the protocol hold in spite of movement-dependent failures, mobile host hardware failures, and also the case where the mobile host is powered off and on again. The *last movement* refers to the last time any mobile host moves from one LR service area to another and switches its serving LR. Note that the maintenance of the location information of a mobile host is independent of the movement of other mobile hosts in the system. Hence, the idea of the *last movement* can be individually applied to each mobile host separately to show that eventually all LRs will have correct location information about each mobile host in the system, and there is no live-lock.

In the following, we state the above properties of the protocol in terms of lemmas and theorems.

Let τ be the time at which the *last movement* has occurred in the system. Let $\mathbf{M}$ be the set of fault-free LRs in the system after some time $t > \tau$. Let N be the total number of mobile hosts that are active in the system.

The following three lemmas and the theorem show that after the last movement, eventually all fault-free LRs will have the same and correct location information for each and every active mobile host in the system.

Lemma 1: At some time $t > \tau$, all fault-free LRs will have the *same* location counter values for each of the active mobile hosts. That is, $\forall(\text{LR}_i, \text{LR}_k) \in \mathbf{M}$, $\forall a(1 \leq a \leq N)$, $\text{LD}_i[a].\text{LC} = \text{LD}_k[a].\text{LC}$.

Lemma 2: At some time $t > \tau$, all the fault-free LRs will have the *same* location information for each of the active mobile hosts. That is, $\forall(\text{LR}_i, \text{LR}_k) \in \mathbf{M}$, $\forall a(1 \leq a \leq N)$, $\text{LD}_i[a].\text{LR} = \text{LD}_k[a].\text{LR}$.

Proof: In order to prove the above two lemmas, consider a mobile host MH_a and assume that, after the *last movement* at time τ, MH_a is served by LR_i. Also, assume that the LC value carried by MH_a before it joined LR_i is LC_a. Consider the following cases.

Case A: $\forall \text{LR}_k \in \mathbf{M}$, $\text{LC}_a \geq \text{LD}_k[a].\text{LC}$

That is, the LC value carried by MH_a is greater than or equal to the LC value any LR has stored in its location directory for MH_a.

In this case, when MH_a joins LR_i, the LC value received by LR_i is also greater than or equal to the LC value stored in all other LRs. Hence, after LR_i increments the LC value, the location information disseminated by it will carry a larger LC value than that stored in any other LR. Since LRs overwrite their location directory with any incoming location information with a larger LC value, each LR will store the same location counter value and the same serving LR_{id} in their location directory for MH_a. Note that any LR_k that claims to be serving MH_a before receiving this location update will send a *check* message to MH_a to verify that MH_a is still in its region. Since MH_a has now registered with LR_i, LR_k will time out on its *check* message and update its location directory with the update information disseminated by LR_i.

Case B: $\exists \text{LR}_k \in M$, $\text{LC}_a < \text{LD}_k[a].\text{LC}$

That is, before registering with LR_i, MH_a carries an LC value that is smaller than the LC value some LR has stored in their location directory. As explained before, this could happen due to three reasons: a movement-dependent failure, hardware corruption, or because the mobile host had been turned off and lost the memory of the LC value. Before MH_a registers with LR_i, let the LC value at LR_i be $\text{LC}_i[a].\text{LC}_{\text{old}}$. Here, there are two cases possible.

Case B.1: $\forall \text{LR}_k \in \mathbf{M}$, $\text{LD}_i[a].\text{LC}_{\text{old}} \geq \text{LC}_k[a].\text{LC}$

That is, the LC value stored in LR_i is not smaller than the LC value stored in any other LR. Since the LC value sent by MH_a is smaller than what LR_i already has, LR_i assumes the LC value from its location directory, increments it, and disseminates the location update. This incremented LC value is strictly larger than the LC value any other LR has stored. This situation is similar to *Case A*. Hence, all LRs will store the same location information about MH_a.

Case B.2: $\exists \text{LR}_k \in \mathbf{M}$, $\text{LC}_i[a].\text{LC}_{\text{old}} < \text{LC}_k[a].\text{LC}$.

That is, the LC value stored in LR_i is smaller than the LC value that some other LR has stored in its location directory.

Here, when LR_i receives the connection request, it will compare the received LC value with that in the location directory, increment the larger of the two LC values, and disseminate the update. Now, there are two cases possible.

Case B.2.1: $\exists \text{LR}_k \in \mathbf{M}$, $\text{LC}_i[a].\text{LC} < LC_k[a].LC$

That is, the incremented LC value is still smaller than that in some other LR. This situation could happen when recent update information from some other service area LR_k has not yet reached LR_i. In this case, when that update information from LR_k arrives, LR_i will increment the incoming LC value and send a *check* message to MH_a. Similarly, when LR_k receives the update information from LR_i, it will also send a *check* message to MH_a. Now, depending on whether MH_a is still in the service area of LR_i, there are two cases possible.

Case B.2.1.1: MH_a still in the service area LR_i.

In this case, MH_a will send an *ackcheck* to LR_i. Then, LR_i will again disseminate the location information of MH_a along with the new LC value. However, LR_k will not receive an acknowledgment from MH_a for its *check* message. Hence, LR_k will time out and update its location directory with the new update information disseminated by LR_i. That is, all LRs will have same location information about MH_a.

Case B.2.1.2: MH_a has moved out of the service area of LR_i.

Now, assume that MH_a has registered with LR_m. Now, when MH_a is registered with LR_m, the LC value sent by MH_a along with the *mmj* message is still incorrect. If the LC value that is stored in LR_m for MH_a is the largest among the LC values for MH_a in all the LRs in the network, then LR_m will use the correct LC value, increment it, and send it back to MH_a as part of *ackmmj*. However, if the LC stored in LR_m is not the largest, then MH_a will still carry an incorrect LC value. In both cases, it is still possible that LR_m will receive a location update from some other LR claiming that it is serving MH_a. In that case, LR_m will send a *check* message to MH_a. Again, if MH_a is still in the service area of LR_m, MH_a will acknowledge the *check* message, and similar to *Case B.2.1.1*, it can be shown that every LR will have the correct location information for MH_a. However, if MH_a again moves out before receiving the *check* message, the correct location information for MH_a cannot be established unless MH_a registers with an LR that has the largest LC value for MH_a or receives an acknowledgment from MH_a. If an LR receives an acknowledgment from MH_a, then, from *Case B.2.1.1*, we can see that every LR will have the same location information about MH_a.

Case B.2.2: $\forall \text{LR}_k \in \mathbf{M}$, $\text{LC}_i[a].\text{LC} \geq \text{LC}_k[a].\text{LC}$

In this case, it is possible that two LRs claim to be serving the same mobile host with the same LC value. Assume that $\text{LC}_i[a].\text{LC} = \text{LC}_k[a].\text{LC}$. Obviously, the update information from LR_k has not reached LR_i yet, but when the updates arrives, LR_i will increment the LC value and send a *check* message to MH_a. LR_k will also send a *check* message to MH_a. Now, from *Case B.2.1*, we can see that all LRs will have the same location information about MH_a. $\diamondsuit$

Lemma 3: At some time $t > \tau$, any fault-free location register LR_i, which becomes the serving LR of an active mobile host MH_a, will have the *correct* location information for MH_a in its location directory. That is, $\text{LD}_i[a].\text{LR} = \text{LR}_i$.

Proof: In our protocol, after a mobile host has registered with an LR (say, LR_i), if an update that claims another LR is serving the same mobile host with a larger or the same LC value arrives, LR_i will always verify whether the mobile host is still in its service area it by sending a *check* message. If the mobile host is still in the service area of MSC_i and if it is active, it will send an *ackcheck* message. Since at some time $t > \tau$ the mobile host has not moved out of LR_i, and it is still active, LR_i will have the *correct* location information for that mobile host.

Theorem 1: At some time $t > \tau$, all fault-free LRs will have the same location counter values and correct location information for each of the active mobile hosts. That is, $\forall \text{LR}_i \in \mathbf{M}$, $\forall a(1 \leq a \leq N)$, $\text{LD}_i[a].\text{LR} = \text{LM}_a$, where LM_a is the location of MH_a after its last movement.

Proof from Lemma 1:

$$\forall (\text{LR}_i, \text{LR}_k) \in M, \quad \forall a(1 \leq a \leq N), \ \text{LD}_i[a].\text{LC} = \text{LD}_k[a].\text{LC}$$

and from Lemma 2

$$\forall (\text{LR}_i, \text{LR}_k) \in B, \quad \forall a(1 \leq a \leq N), \ \text{LD}_i[a].\text{LR} = \text{LD}_k[a].\text{LR}.$$

But, from Lemma 3, for a mobile host MH_a, $\text{LD}_i[a].\text{LR} = \text{LR}_i = \text{LM}_a$.

Hence, $\forall (\text{LR}_i, \text{LR}_k) \in M$

$$\text{LD}_i[a].\text{LC} = \text{LD}_k[a].\text{LC}$$

and

$$\text{LD}_i[a].\text{LR} = \text{LD}_k[a].\text{LR} = \text{LR}_i = \text{LM}_a.$$

Since the location update of a mobile host is independent of the movement of the other mobile hosts, all LRs will have same and correct location information for each of the active mobile hosts in the system.

Theorem 2: After time $t \geq t'$, where $t' > \tau$, no protocol specific messages are exchanged in the network, and only the periodic *beacon* signal is emitted on the wireless channel.

Proof (proof by contradiction): In the protocol, the update messages are disseminated in the network by diffusion. Upon receiving an update, an LR checks whether it is new information. If so, then the LR forwards the update to all of its neighbors except the one from which it received the update.

Now assume that after time t' there is an update message about the location of a mobile host MH_a being exchanged in the wired network. Any LR will forward an incoming update message only if the LC value associated with the message is larger than what the LR has stored in its location directory. Let the maximum delay in the wired network be δ, and let t_{ack} be the maximum delay in receiving an acknowledgment from a mobile. Then, assuming that the time out on acknowledgment $t_{\text{out}} < \delta$, from *Theorem 1* we know that after time $t' = \tau + 2\delta + t_{\text{ack}}$ all LRs will have the same and correct location counter value and location information of each and every active mobile host. Since the location update of a mobile host is disseminated only by the LR that is currently serving it, the LC value associated with the update could not have been larger than what all LRs have stored after time t'. And, if the LC value of the update message were smaller, then this message would have been ignored by all the MSCs, and no further propagation would be carried out.

However, the case of an inactive mobile host is different. Now, if that mobile host has become inactive after time τ (i.e., after all LRs have the same and correct location information), then from the previous paragraph, we know that no protocol-specific messages will be exchanged in the network. But, if that mobile host has become inactive before time $t < \tau$, then all LRs may not have the correct location information for that mobile host. However, after time $t > \tau$, all LRs will have the update information corresponding to the largest LC value in their location directory. If more than one LR disseminates the update with the same large LC value, then all LRs will store all those update information in their location directory. If they get this same information again, that update will not be propagated further. Hence, there is no update information exchanged in the network after some time $t > t'$.

III. Location Management in Large Networks

The location information is fully distributed in the basic scheme so as to reduce the call setup time and tolerate node failures. However, the fully distributed scheme does not scale well for large networks due to the overhead in location information dissemination and storage. In addition, the usefulness of the benefits of the fully distributed scheme—call setup time and fault tolerance—diminishes as the network becomes larger. In this section, we discuss how the basic scheme can be tailored for a large network so that the location management overhead is contained. We discuss two schemes: in the first one, the location information is fully disseminated but the LRs selectively store it; the second one uses a hierarchical arrangement of LRs so that the dissemination itself is limited.

A. Caching

Caching scheme is similar to PCS architecture. Each mobile host is associated with an HLR. But, instead of VLR, the caching scheme uses LRs for the dual function of visitor location registry and lookup directory for nonlocal mobile hosts. When a call originates for a mobile, the current location of the mobile is found by first querying the local LR. If the location information is not cached in the local LR, only then is the HLR contacted.

When the mobile moves from one LR region to another, this location information is updated at the HLR and disseminated to all the LRs in the network. Other LRs selectively cache a loca-

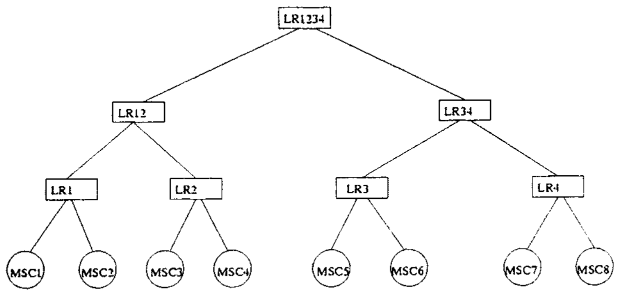

Fig. 6. Conceptual diagram showing the hierarchical arrangement.

tion information. The decision to cache the location of a mobile host could be made based on any of the schemes discussed in [8] and [11]. In the schemes presented in [8] and [11], the caching decision is based on the local call mobility ratio, which is the ratio between the number of calls to a mobile host from an LR area and the number of times the mobile host has changed its location as seen by that LR. The LR sees changes in the location of a mobile only when a call originates for that mobile from the LR area. Hence, even if the mobile host changed several LR areas between two consecutive calls from an LR area, the LR will count the mobility of that mobile host as just one movement between its consecutive calls. Since the location information is fully disseminated in our scheme, all LRs can keep track of the true mobility of any mobile host. Hence, LRs can cache the location information efficiently and yield better hit rates. Also, if location information is not disseminated (as in [8] and [11]), the cache is more likely to have stale entries compared to our scheme. If an LR uses this stale entry to contact a mobile host, it will send a connection request to a wrong LR first and wait for the reply from that LR before it can contact the HLR of that mobile to receive the current location. If the hit rates are poor, the cost of contacting a wrong LR negates the gain of caching the location information. But a full dissemination yields better hit rates. The connection establishment time will be low in the caching scheme.

1) Caching Location Information for the First Time: Even though an LR caches the location information of only a limited number of mobile hosts, it receives the location updates of every mobile host in the network. But, when an LR decides to cache the location information of a mobile for the first time, it has to contact the HLR of this mobile to get the most recent location information. The LR cannot simply use the update information it received on the network to cache the location of a mobile for the first time. This is because of the possibility that the information received over the network may be old; the LR will not be able to determine that a location information is old or new because it does not know the current LC value associated with the mobile. The location information that is subsequently received over the network can then be used to *validate* the cached information. In this case, the LR can make a determination as to whether the received information is new or not by comparing the location counter value of the already stored information with that of the newly received information.

While an LR is waiting on a reply from an HLR to which it had sent a request asking for the location information of a mobile, and receives a location update disseminated for this mobile, it should buffer this update information before the reply from the HLR is received. Once the reply is received, the location information received from the HLR can be compared with the location information stored in the buffer to determine which one is more recent. This information can then be cached.

B. Hierarchy

Although in the caching scheme location information about any subscriber can be cached by any LR in the network, the protocol requires that new location information about all mobiles be disseminated to all the LRs in the network. As the network expands, this requirement could inflict heavy traffic on the signaling network. One way to reduce the signaling traffic is to arrange the LRs in a hierarchy similar to the one proposed by Wang [21]. In the hierarchical scheme, when a subscriber roams away from his/her home area to a new area, an address chain is created from the new area up to the common parent of the home area and the new area. Now we will see how our generic scheme can be modified to fit the hierarchical structure proposed by Wang.

Fig. 6 shows the conceptual arrangement of the LRs in the proposed scheme. Only the LRs in level one are directly connected to the MSCs. The LRs in level 1 are connected to LRs in level 2, LRs in level 2 are connected to LR in level 3, and so on. The LRs under the same parents are clustered together (i.e., [LR_1, LR_2] forms a level-1 cluster, [LR_3, LR_4] forms another level-1 cluster, and [LR_{12}, LR_{34}] forms a level-2 cluster). Now, we modify the location directory maintenance protocol described in Section II to fit the proposed hierarchical arrangement of LRs.

1) *Movement of the MH Is Across the LRs in Level 1.* As described earlier, if the movement of the MH is within an LR area in level 1, then no update information needs to be disseminated. The LR, where the movement occurs, just updates its location directory with the current MSC serving the MH. However, if the movement is across the LR area in level 1, and if the new LR that the MH has moved to is also under the same parent LR as the one it moved away from, then the location information of the MH is disseminated to all the LRs in that level-1 cluster. The parent LR will also update the new location information. The location information is not propagated any further.
2) *Movement of the MH Is Across the LR Area in Level 2.* Assume that MH has moved from MSC_4 to MSC_5. When the

MH joins LR_3 through MSC_5, LR_3 will check its location directory to find out the last LR that served the MH. Since the MH has moved in from another cluster, LR_3 will not have any cache entry for the MH. Hence, it will forward the *mmj* request to its parent LR (i.e., LR_{34}), which has the previous location of the MH (i.e., LR_{12}). Once LR_{34} receives the *mmj* request, it will contact LR_{12} and receive the new LC value. Then, LR_{34} will compare the LC it received from LR_{12} and the one that arrived with the *mmj* message, increment the larger of the two, and send it back to LR_3. LR_3 will then send an *ackmmj* back to the MH along with the new LC value. MSC_3 will disseminate the new location information to all other LRs (i.e., LR_4) in its cluster. In the meantime, LR_{34} will disseminate the new location information to all other LRs in its cluster (i.e., LR_{12}).

While the MH is in MSC_4's area, all the LRs under LR_{12} (i.e., LR_1 and LR_2) would have cached the location information of that MH. This cached information has to be removed when the MH establishes connection with LR_3. This is done by LR_{12} by sending an *invalidate* message to all its child LRs when it receives the update information from LR_{34}. The structure of the *invalidate* message is $MH_{id}|LC$. Upon receiving the *invalidate* message, LRs will remove the location directory entry for the MH from their cache.

IV. Conclusions

We considered the problem of maintaining location information about mobile subscribers in a PCS network. We presented a generic protocol for location management where location information was replicated and new location information of the mobile was disseminated throughout the network. Fully distributing the location management reduces the call setup time and avoids single point of failure that exists in centralized location management like IS-41. Fully distributing the location management is not an efficient solution for large networks due to large number of location updates that need to be disseminated and processed. For large networks, we discuss two alternative solutions that limit the location updates messages while still keeping low call setup time.

An analytical comparison of the performance of fully distributed location management with IS-41 protocol confirms our premise that when the call arrival rate is high and mobility is low, overall location management cost is lower using fully distributed location management than IS-41 [19]. It shows that for a network of 25 service areas (i.e., HLRs), if the average residence time of a user in a service area is about 10 h and the call arrival rate is as low as one call every two hours, the cost of fully distributed location management is lower than IS-41. For the same values, if the network size is 100 service areas, then fully distributed location management is cost efficient only if the call arrival rate is four calls per hour or higher. An analytical comparison of caching approaches based on IS-41 [12] and the one proposed in this paper (see Section III-A) can be found in [17]. It shows that for a network of 25 service areas, our caching scheme is more cost efficient than an IS-41-based caching scheme when the call arrival rate is higher than two calls per hour.

It is assumed here that location information is distributed by flooding the network so that the protocol is tolerant to node failures. However, if the nodes do not fail, then location information can be efficiently distributed by using a spanning tree rooted at the node that initiates the location update. In this way, the overall cost of fully distributed location management can be significantly reduced.

The protocol has been simulated using the process-oriented simulation language CSIM. The details of the implementation of the basic protocol can be found in [14]. We have also implemented the caching and hierarchical approached proposed in this paper in the simulation. For comparison, we have simulated IS-41, as well as the IS-41-based caching approach. We are currently comparing the performance of these different location management schemes.

References

[1] I. F. Akyildiz and J. S. M. Ho, "On location management for personal communications networks," *IEEE Commun. Mag.*, Sept. 1996.

[2] B. R. Badrinath *et al.*, "Locating strategies for personal communication networks," in *Proc. IEEE Globecom Workshop Networking of Personal Communication*, Dec. 1993.

[3] A. Bar-Noy and I. Kessler, "Tracking mobile users in wireless networks," *IEEE Trans. Inform. Theory*, 1993.

[4] P. Bhagwat and C. Perkins, "A mobile networking system based on Internet Protocol (IP)," in *Proc. Usenix Symp. Mobile and Location-Independent Computing*, Aug. 1993.

[5] Y. Cui, D. Lam, J. Widom, and D. C. Cox, "Efficient PCS call setup protocols," in *Proc. 17th Annu.IEEE Joint Conf. Computer Communications (Infocom '98)*, San Francisco, CA, Mar. 1998, pp. 728–736.

[6] M. D. Gallagher and R. A. Snyder, *Mobile Telecommunications Networking with IS-41*. New York: McGraw-Hill, 1997.

[7] EIA/TIA/IS-41, "Cellular radiotelecommunications intersystem operations,", Dec. 1990.

[8] H. Harjono, R. Jain, and S. Mohan, "Analysis and simulation of a cache based auxiliary location strategy for PCS," in *Proc. IEEE Conf. Networks for Personal Communications*, Mar. 1994.

[9] J. S. M. Ho and I. F. Akyildiz, "Local anchor scheme for reducing location tracking costs in pcns," *IEEE/ACM Trans. Networking*, Oct. 1996.

[10] Y.-B. Lin, "Determining the user locations for personal communications networks," *IEEE Trans. Veh. Technol.*, vol. 43, pp. 466–473, 1994.

[11] R. Jain, Y.-B. Lin, C. Lo, and S. Mohan, "A caching strategy to reduce network impacts of PCS," *IEEE J. Select. Areas Commun.*, vol. 12, pp. 1434–1444, Oct. 1994.

[12] ——, "A forwarding strategy to reduce network impacts of PCS," in *Proc. IEEE Infocom*, Apr. 1995.

[13] P. Krishna, N. H. Vaidya, and D. K. Pradhan, "Location management in distributed mobile environments," in *Proc. 3rd Int. Conf. Parallel and Distributed Information Systems*, Sept. 1994.

[14] K. C. Lee, "A test-bed for mobile environment simulation," master's thesis, Northeastern Univ., 1996.

[15] C. N. Lo, R. S. Wolff, and R. C. Bernhards, "An estimate of network database transaction volume to support personal communication services," *Proc. IEEE 1st Int. Conf. Universal Personal Communications*, pp. 236–241, 1992.

[16] S. Rangarajan, K. Ratnam, and A. T. Dahbura, "A fault-tolerant protocol for location directory maintenance in mobile networks," in *Proc. 25th Int. Symp. Fault-Tolerant Computing*, June 1995.

[17] K. Ratnam, I. Matta, and S. Rangarajan, "Analysis of caching-based location management in personal communications networks," in *Proc. IEEE 7th Int. Conf. Network Protocols (ICNP99)*, Nov. 1999.

[18] ——, "A fully distributed location management scheme for large PCS," in *Proc. 5th IEEE Symp. Computer and Communications (ISCC 2000)*, July 2000.

[19] K. Ratnam, "Efficient location management and packet delivery in mobile communications networks," Ph.D. dissertation, Northeastern University.

[20] N. Shivakumar, J. Jannink, and J. Widom, "Per-user profile replication in mobile environments: Algorithms, analysis, and simulation results," *ACM/Baltzer J. Mobile Networks and Applications*, vol. 2, no. 2, pp. 129–140, 1997.

[21] J. Z. Wang, “A fully distributed location registration strategy for universal personal communication system,” *IEEE J. Select. Areas Commun.*, pp. 850–860, Aug. 1993.

Karunaharan Ratnam recieved B.Sc.Eng. degree in electrical engineering from the University of Peradeniya, Sri Lanka, in 1990 and the M.S. degree in electrical and computer engineering from the University of Alabama in Huntsville in 1993.

He joined GTE Laboratories, Waltham, MA, in 1995 and currently is a Principal Member of Technical Staff in the Backbone Technologies Department. Dr. Ratnam is currently working on a scalable architecture for large-scale deployment of voice and other enchanced services over the Internet. His research interests are quality of service, voice over IP, and mobile computing.

Sampath Rangarajan recieved the bachelor’s degree (with honors) in electronics and communication engineering from the University of Madras, India, in 1984. He received the M.S. degree in electrical and computer engineering and the Ph.D. degree in computer sciences from the University of Texas at Austin in 1987 and 1990, respectively.

Currently, he is a Member of Technical Staff in distributed software at Bell Laboratories, Murray Hill, NJ. Before that, he was an Assistant Professor in the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA (1992–1996) and a Research Associate with the University of Maryland Institute for Advanced Computer Studies (1990–1992). His research interests are in the areas of distributed computing, fault-tolerant computing, and networking.

Anton T. Dahbura (F’96) recieved the B.S.E.E., M.S.E.E., and Ph.D. degrees in electrical engineering and computer science from the John Hopkins University, Baltimore, MD, in 1981, 1982, and 1983, respectively.

In 1983, he joined the Computer System Research Laboratories, AT&T Bell Laboratories, Murray Hill, NJ, and conducted research on fault detection and diagnosis algorithms. From 1990 to 1996, he was Research Director of the Motorola Cambridge Center, Cambridge, MA, responsible for research activities in the area of parallel and distributed computing systems. From 1996 to 1998, he was a Consultant to Digital Equipment Corporation (now Compaq). His research interests include dependable and mobile computing, combinational optimization algorithm, and applications of NII. He is a member of IFIP Working Group 10.4 (Dependable Computing and Fault Tolerance). He is the author of more than 50 technical papers and has received three U.S. patents.

Dr. Dahbura is a Fellow of the IEEE Computer Society and a member of the ACM (SIGACT).