Forwarding Pointers for Efficient Location Management in Distributed Mobile Environments *
(Preliminary Version)

P. Krishna † N. H. Vaidya D. K. Pradhan
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112
E-mail: {pkrishna,vaidya,pradhan}@cs.tamu.edu
Fax: (409) 862-2758
Phone: (409) 862-2599

September, 1994

Technical Report # 94-061

Abstract

Location management consists of location update and search protocols. Trade-off in location management is between the cost of searches and the cost of updates. The goal of an "efficient" location management strategy is to reduce the search and update costs. In this paper, we show that, using forwarding pointers in addition to home location servers help in providing efficient location management. It is shown that using forwarding pointers reduces the network load during updates, however, use of forwarding pointers may increase the search cost due to long chain lengths. We present two heuristics, namely, *movement-based* and *call-based*, to limit the length of the chain of forwarding pointers. It was observed that for most of the call-mobility patterns *call-based* heuristic performed better than the *movement-based* heuristic.

A *search-update* occurs after a successful search, when the location information corresponding to the searched mobile host is updated at some hosts. This paper proposes various strategies for search-updates. It was noticed that performing *search-updates* significantly reduced the search costs with very little cost to pay for updates (upon moves and searches).

*Research reported is supported in part by AFOSR
†Direct all correspondence to P. Krishna.

1 Introduction

Location management is one of the most important issues in distributed mobile computing. Location management consists of location *updates*, *searches* and *search-updates*. An *update* occurs when a mobile host changes location. A *search* occurs when a host wants to communicate with a mobile host whose location is unknown to the requesting host. A *search-update* occurs after a successful *search*, when the requesting host updates the location information corresponding to the searched mobile host.

Integrated voice and data application will be used by millions of people often moving in a very heavy urban traffic conditions. Some of the proposed location management schemes assume location updates upon crossing cell boundaries. This is not a problem in a lightly loaded network, but it may be a significant problem in a high density environment [3]. In [3] it has been shown that if location updates were to occur on each cell crossing the resulting signalling load will have a major impact on the load of the network. The additional signalling traffic on the SS7 signalling system (capacity of 56 Kbps) is expected to be 4-11 times greater for cellular networks than for ISDN and 3-4 times greater for future personal communication networks (PCN) than for cellular networks. The signalling load due to updating alone contributes 70% additional load. Thus location update will become a major bottleneck at the switches (such as SS7) and mechanisms to control the cost of location update are needed.

In this paper we present schemes that use forwarding pointers to facilitate efficient location management. Use of forwarding pointers to track decentralized objects was proposed by Fowler [7]. In our earlier work, we discussed location management schemes using forwarding pointers for network architectures consisting of a hierarchy of location servers and without designated home location servers [1]. In this paper we deal with network architectures in which there is a home location server (HLS) for every mobile host. Location management with home location servers have been earlier proposed in the literature [6, 11]. Badrinath et. al. examined strategies that reduced search costs and at the same time control the volume of location updates by employing user profiles [6]. The user profiles were used to create partitions. When the user crosses partitions, does the update takes place. However, user profiles are not always available a priori.

There are existing standards for carrying out location management using home location servers (HLS), e.g., Electronic Industry Association/Telecommunication Industry Association (EIA/TIA) Interim Standard 41 (IS-41)[1] [11]. However, these schemes are not efficient, due to increase in network load and location management costs. To overcome these drawbacks, this paper presents location management strategies using forwarding pointers in addition to HLS. We

[1] The terminology used in IS-41 literature is slightly different. IS-41 uses home location register (HLR) and visitor location register (VLR) databases. However, the information maintained in the HLR is same as what is maintained in HLS, the information maintained in the VLR is maintained in the various location servers.

present two heuristics to limit the length of the chain of forwarding pointers. We show that significant improvement can be obtained using forwarding pointers in addition to HLS. The paper also presents strategies to perform *search-updates*. It was noticed that performing *search-updates* significantly reduced the search costs with very little cost to pay.

2 Network Architecture

In this section we present a network architecture for a distributed system with mobile hosts [9, 10, 11]. The static network comprises of the fixed hosts and the communication links between them. Some of the fixed hosts, called *mobile support stations* (MSS) [2] are augmented with a wireless interface, and, they provide a gateway for communication between the wireless network and the static network. Due to the limited range of wireless transreceivers, a mobile host can communicate with a *mobile support station* only within a limited geographical region around it. This region is referred to as a mobile support station's *cell*. As shown in Figure 1, a characteristic feature of cellular radio is the association of the hexagonal with each radio cell surrounding a *mobile support station*. In practice. the cell coverage area may not be of any regular shape. The geographical area covered by a *cell* is a function of the medium used for wireless communication. Currently, the average size of a cell is of the order of 1-2 miles in diameter. As the demand for services increase, the number of cells may become insufficient to provide the required grade of service. *Cell splitting* can then be used to increase the traffic handled in an area without increasing the bandwidth of the system. In future, the cells are expected to be very small (less than 10 meters in diameter) covering the interior of a building [2]. A mobile host communicates with one *mobile support station* (MSS) at any given time. MSS is responsible for forwarding data between the mobile host (mh) and static network.

Cells are grouped into *registration areas*. There is a location server in each registration area. Each mobile host is assumed to be permanently registered to a particular registration area. The location server of that registration area is called the *home location server* (HLS) for the mobile host. This association of a host with a particular *home location server* is fully replicated across the whole network. The *home location server* is responsible keeping track of the location information of the mobile host. The location server is also responsible for maintaining location information of the mobile hosts currently residing (visitors) in a cell in its registration area. Location servers of different areas are connected to *regional switching centers*. These switching centers perform the message routing functions.

When a mobile host is engaged in a call or data transfer, it will frequently move out of the coverage area of the mobile support station it is communicating with, and unless the call is passed on to another cell, it will be lost. Thus, the task of forwarding data between the static

[2] Mobile support stations are sometimes called *base stations*.

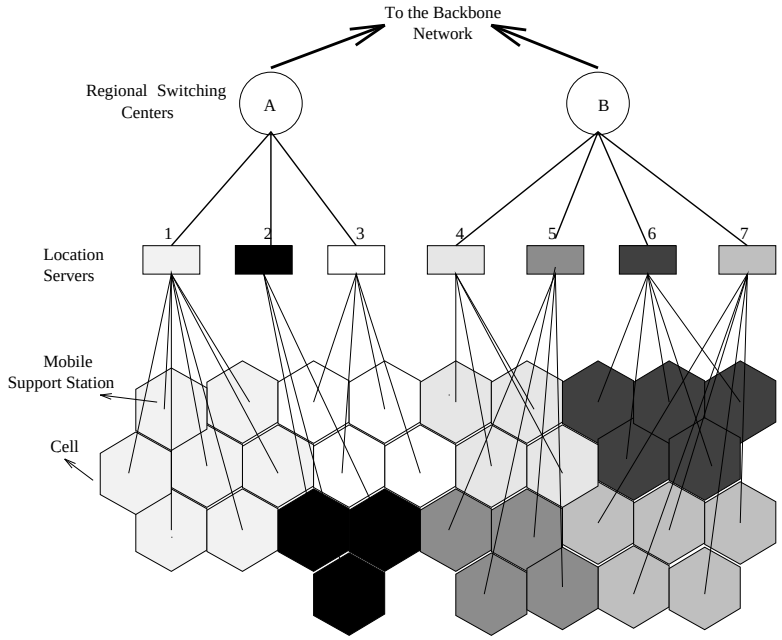

Figure 1: Network Architecture

network and the mobile host must be transferred to the new cell's *mobile support station*. This process, known as *handoff*, is transparent to the mobile user. The handoff can be network-controlled or mobile-assisted [9, 10]. In network-controlled handoffs, the system continuously monitors the signals received from the mobile hosts engaged in calls, checking on signal strength and quality. When the signal falls below a preset threshold, the system will check whether any other MSS can receive the mobile host at a better strength. In mobile-assisted handoffs, the mobile host during its non-active periods is able to measure the signals from adjacent MSSs, report back to the network, and aid in the decision to execute handoff. Therefore, handoff helps to maintain an end-to-end connectivity in the dynamically reconfigured network topology.

3 Location Management

3.1 Home Location Servers (HLS)

Location management with home location servers is being used in current personal communication systems standards proposals such as IS-41 [11]. In this section, we will first present the basic scheme for updates and searches that is being currently used in IS-41. We will then present the drawbacks of these schemes. Later in this section we will present the modified location management scheme using forwarding pointers.

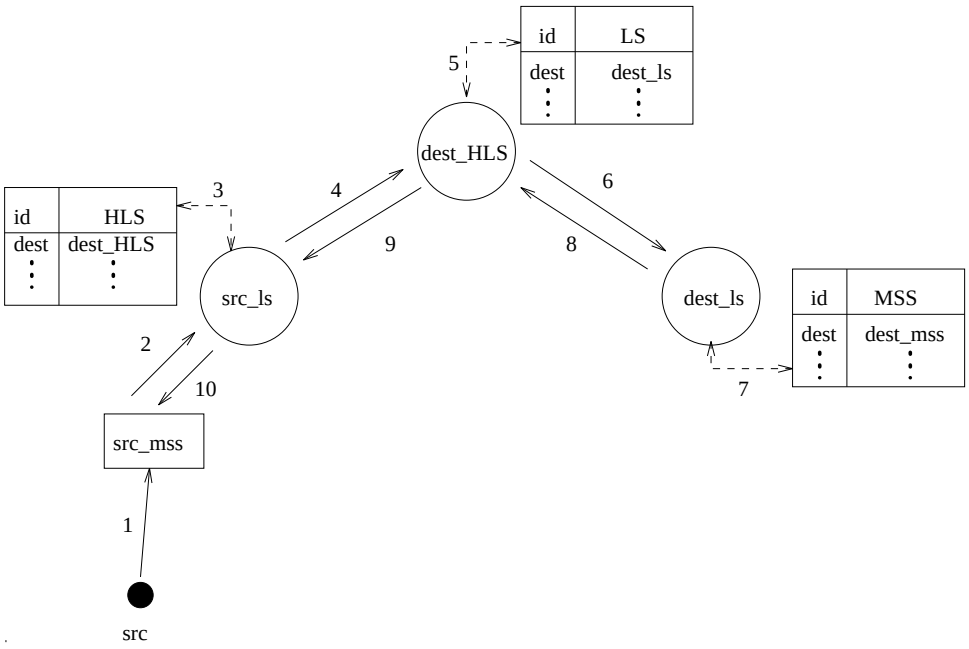

Figure 2: Search using HLS

3.1.1 Location Updates

Location updates take place only when the mobile host (mh) enters a new registration area. Let the old registration area be old, and the new registration area be new. Upon entering new, the mobile host (mh) informs its entry to the new mobile support station, which in turn forwards the information to the location server of registration area new, named new_ls. The new_ls in turn sends a message to HLS of mh notifying it about the new location of mh. The HLS updates the location information of mh (i.e., changes the current registration area to new), and sends the host information (user profile etc.) to new_ls. The HLS also sends a message to the location server of registration area old, named old_ls, requesting it to delete any host information stored at old_ls. The location information of the mobile host is then purged from old_ls. After which, old_ls sends a confirmation message to HLS.

3.1.2 Location Searches

Let us suppose that a mobile host src wants to communicate with another mobile host $dest$. Let src_mss be the MSS of the cell in which host src currently resides. When src wants to communicate with a destination mobile host $dest$, it has to first determine the location of $dest$. Host src sends a location query message to src_mss (step 1 in Figure 2). The src_mss, in turn forwards the query to the location server (say src_ls) in its registration area (step 2 in Figure 2). The location server src_ls checks whether $dest$ is in its registration area. If $dest$ is in its registration

area, then it returns the location information. Otherwise, src_ls looks up its database to determine the HLS of $dest$ (say $dest\_HLS$) based on the identification of $dest$ (step 3). The location server src_ls then forwards the location query message to $dest\_HLS$ (step 4). The $dest\_HLS$ maintains the identification of the location server (say $dest\_ls$) of the registration area in which the last location update of $dest$ took place (step 5). The $dest\_HLS$ queries the $dest\_ls$ for the cell location (identifier of the MSS, $dest\_mss$) of $dest$ (step 6). Upon getting a reply from $dest\_ls$ (steps 7 and 8), $dest\_HLS$ returns the current location of $dest$ to src_ls, which, in turn forwards it to src_mss (steps 9 and 10). The location information is nothing but the address of the MSS (named $dest\_mss$) of the cell in which the mobile host $dest$ currently resides. Thereafter, the call is set up between src and $dest$ via src_mss and $dest\_mss$.

3.1.3 Drawbacks

- Increase in network traffic when the host crosses registration areas very frequently : This can be due to a very mobile host going in some specific direction, or because the host moves in and out of a registration area such that there are lot of registration area boundary crossings. As every registration area crossing causes a location update at the host's HLS, this scheme increases the network traffic.
- Inefficient location management : Updates on each registration area crossing is useful, if the user is being called frequently, to reduce the search costs (i.e., call set-up time). However, if the user is not being called frequently, regular updates are not necessary. In such scenarios, regular updates lead to inefficient location management.

3.2 HLS and Forwarding Pointers

In this section we will discuss schemes to avoid the drawbacks of the above location management strategy. We use forwarding pointers in addition of HLS to assist in location management. Our main goal is to avoid the increase in network traffic due to location updates at the HLS. We achieve this by not updating the location information after every registration area crossing. Instead, forwarding pointers are maintained at the location servers of the registration areas visited. To avoid an increase in the cost of location searches, the length of the chain of forwarding pointers should not be too long. Therefore, the HLS is updated after certain interval of time which is determined based on heuristics discussed later. In the following, we present the proposed scheme.

3.2.1 Location Updates

Whenever a mobile host leaves a registration area, say old, and enters a new registration area, say new, the host information (e.g., user profile, current length of the chain of forwarding pointers

etc.) is transferred from old_ls to new_ls, where, $old\_ls(new\_ls)$ is the location server for $old(new)$ registration area. In addition, a forwarding pointer is created for the mobile host at old to point to new.

Update at the HLS do not take place for every registration area crossing, but after an interval of time, the time interval being determined using the heuristics explained below. The procedure to update the HLS is same as the update in the previous section. After the update, HLS will know the present location of the mobile host, and no forwarding pointers need to be traversed (until the mobile host moves from its present registration area).

Heuristics

- Movement-based : HLS update takes place when the number of registration area crossing is M. A similar heuristic was proposed in [8] for a different network architecture.
- Call-based : HLS update takes place whenever a call arrives. Therefore, HLS update will take place whenever a location search takes place. A similar approach was proposed in our earlier work [1] for a different network architecture.

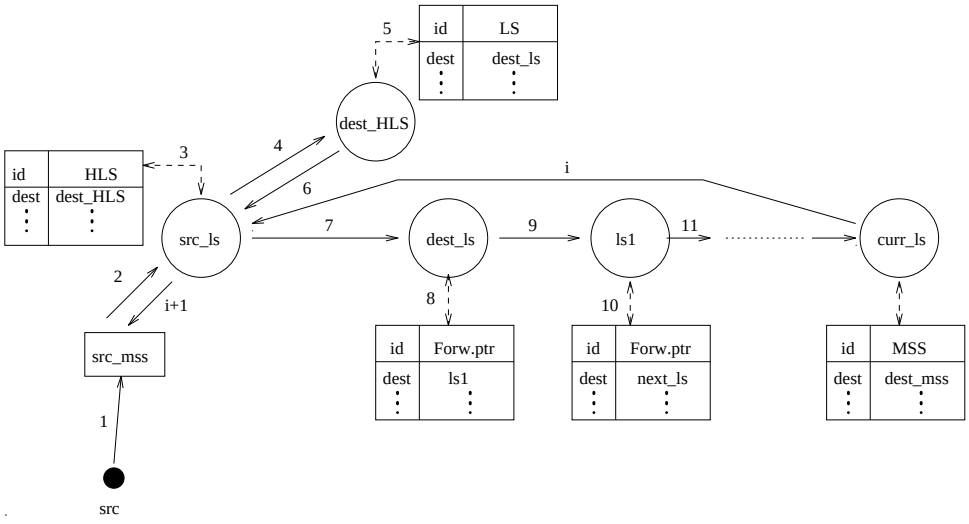

Figure 3: Search using HLS and Forwarding Pointers

3.2.2 Location Searches

Let us suppose that a mobile host src wants to communicate with another mobile host $dest$. Please refer Figure 3 for this discussion. Search process, until the location query is forwarded to the home location server of $dest$, $dest\_HLS$, is same as previously explained (steps 1-4 in Figure 3). $Dest\_HLS$ returns the address of the location server ($dest\_ls$) of the registration area in which the

mobile host was residing when the last HLS update took place (steps 5 and 6). Upon receiving the address of $dest\_ls$, src_ls traverses the chain of forwarding pointers originating at $dest\_ls$ (step 7). Each location server on the chain look up their database to determine the next location server on the chain, and forwards the location query to it (e.g., steps 8 and 9 for $dest\_ls$). This continues till the end of the chain is reached (say $curr\_ls$ is the location server). The host $dest$ will be located in a cell in the registration area of $curr\_ls$. The location server $curr\_ls$ returns the current cell location to src_ls, which in turn forwards it to src_mss (steps i and $i+1$). Thereafter, the call is set up between src and $dest$ via the src_mss and $dest\_mss$.

4 Performance Analysis

In this section we analyze the modified location management scheme using forwarding pointers and compare it with the basic scheme using only HLS. We will then compare the performance of the two heuristics presented in this paper.

Call arrivals from within the registration area will not involve the home location servers. Moves within the registration area will also not involve any home location server interaction. Therefore, we analyze the performance of the location management schemes for calls which arrive from outside the current registration area (hereafter, we refer them simply as call arrivals), and for moves which are registration area crossings.

- r = ratio of mean rate of call arrivals and mean rate of registration area crossings, i.e., r call arrivals per registration area crossing. We call this the call-mobility ratio. This is little different than definition of call-mobility ratio in the existing literature [6].
- C_u = cost of a location update per registration area crossing using only HLS.
- C_s = cost of a location search using only HLS.
- $C_t = C_u + rC_s$ = total cost of update and searches per registration area crossing, when using only HLS.
- C'_u, C'_s, C'_t are the update, search and total cost, respectively, using HLS and forwarding pointers. $C'_t = C'_u + rC'_s$.
- $\text{cost}(x \rightarrow y)$ = cost of processing and sending a message from x to y.
- $h_{(x,y)}$ = number of hops between nodes x and y.
- W = cost of processing and sending a location query/reply message over wireless link.

- S = cost of processing and sending a location query/reply message between MSS and the LS within a registration area.
- $B(h)$ = cost of processing and sending a location query/reply message between any two LSs over the backbone network; h is the number of hops between the two LSs.

Let a mobile host mh move from a registration area to a new registration area, and, let the corresponding location servers be *old* and *new* respectively. Let the home location server of mh be HLS. From the location update scheme presented in the previous section, we can obtain,

$$\begin{aligned} C_u &= \text{cost}(mh \rightarrow new) + \text{cost}(new \rightarrow HLS) + \text{cost}(HLS \rightarrow old) \\ &\quad + \text{cost}(old \rightarrow HLS) + \text{cost}(HLS \rightarrow new) \\ &= W + S + 2B(h_{(new,HLS)}) + 2B(h_{(HLS,old)}) \end{aligned}$$

Let a mobile host src call another mobile host $dest$. Let the location servers of the registration areas in which src and $dest$ are currently residing be src_ls and $dest\_ls$ respectively. Let the home location server of $dest$ be HLS. From the Figure 2, we can obtain,

$$\begin{aligned} C_s &= \text{cost}(src \rightarrow src\_mss) + \text{cost}(src\_mss \rightarrow src\_ls) + \text{cost}(src\_ls \rightarrow HLS) \\ &\quad + \text{cost}(HLS \rightarrow dest\_ls) + \text{cost}(dest\_ls \rightarrow HLS) + \text{cost}(HLS \rightarrow src\_ls) \\ &\quad + \text{cost}(src\_ls \rightarrow src\_mss) \\ &= (W + 2S) + 2B(h_{(src\_ls,HLS)}) + 2B(h_{(dest\_ls,HLS)}) \end{aligned} \quad (1)$$

Now we will evaluate the schemes with forwarding pointers. Let,

- k = length of the chain of forwarding pointer when a search takes place. This is the number of forwarding pointers traversed before locating the destination host.
- K_u = length of the chain of forwarding pointers when the proposed scheme performs an update. M_u is number of registration area crossings when the length of the chain is K_u. $M_u = (K_u - 1)$.
- F = cost of creating a forwarding pointer between two location servers. This cost includes the cost of sending a message, and the creation of a data structure at a location server. The cost of traversing a forwarding pointer includes the cost of looking up the data structure at the location server, and sending a message. The cost of traversing a forwarding pointer is also assumed to be F.

Since the length of the chain of forwarding pointer is K_u when an update takes place, K_uF is the total cost of creating forwarding pointers before an update takes place. Therefore the average

update cost for the scheme with forwarding pointers is as follows :

$$C'_u = (1/M_u)(K_u F + C_u)$$

From Figure 3 we can obtain the search cost as follows :

$$\begin{aligned} C'_s \;=\; & \text{cost}(src \rightarrow src\_mss) + \text{cost}(src\_mss \rightarrow src\_ls) + \text{cost}(src\_ls \rightarrow HLS) \\ & + \text{cost}(HLS \rightarrow src\_ls) + \text{cost}(\text{traversing the forwarding pointers to } curr\_ls) \\ & + \text{cost}(curr\_ls \rightarrow src\_ls) + \text{cost}(src\_ls \rightarrow src\_mss) \end{aligned}$$

We make a conservative estimate of the cost($curr\_ls \rightarrow src\_ls$). In the worst case, this cost will be equal to kF (the cost of traversing the forwarding pointers). Therefore,

$$C'_s = (W + 2S) + 2B(h_{(src\_ls,HLS)}) + 2kF \tag{2}$$

Let $X = W + 2S$. Then, from equations (1) and (2), we get,

$$C'_s = X/2 + C_s/2 + 2kF$$

The maximum search cost will occur when the length of the chain is maximum possible. This is the length of the chain just before an update, i.e., K_u. Therefore,

$$\max_k(C'_s) = X/2 + C_s/2 + 2K_u F$$

If we assume that the uniform distribution for call arrival times between two registration area crossings, average value of k will be $K_u/2$. Therefore, the average search cost is as follows :

$$C'_s = X/2 + C_s/2 + K_u F$$

We will determine the relative total cost (C'_t/C_t) and relative search cost (C'_s/C_s).

4.1 Analysis

$$\frac{C'_t}{C_t} = \frac{\frac{r}{2}(X + C_s + 2K_u F) + \frac{1}{M_u}(C_u + K_u F)}{rC_s + C_u} \tag{3}$$

We are analyzing the case when the caller and the destination host are always in different registration areas, and, also that all moves are registration area crossings. Thus, a move will always create a forwarding pointer, and a search will involve traversing forwarding pointers. We assume that the cost of HLS interactions is the dominant cost in C_s and C_u. Since, C_u and C_s have the same number of HLS interactions, we assume $C_u = C_s$, and, $X \ll C_s$. The cost of creating a

forwarding pointer and traversing a forwarding pointer is less than the cost of updates and searches. It is assumed to be a fraction of the cost of update and searches, i.e., $F = \alpha C_u$, where, $\alpha \leq 1$.

$$\frac{C'_t}{C_t} = \frac{(\frac{r}{2} + \frac{1}{M_u})(1 + \alpha K_u)}{r+1} + \frac{\alpha r K_u}{r+1} \tag{4}$$

$$\frac{C'_s}{C_s} = \frac{1}{2}(1 + 2\alpha K_u) \tag{5}$$

As explained in the previous section, there are two heuristics to decide when to perform a location update at the HLS. They are movement-based heuristic and call-based heuristic.

4.1.1 Movement-based Heuristic

In such a heuristic, the location information at the HLS gets updated every M registration area crossings. The length of the chain of forwarding pointers will be $M-1$. Thus, replace $M_u = M$, and $K_u = M_u - 1$ in the above equations.

$$\frac{C'_t}{C_t} = \frac{1}{2M}(\frac{(1+Mr)(1+2\alpha(M-1))+1}{r+1}) \tag{6}$$

$$\frac{C'_s}{C_s} = \frac{1}{2}(1 + 2\alpha(M-1)) \tag{7}$$

4.1.2 Call-based Heuristic

A HLS update takes place upon every call. Since, there are r calls per registration area crossing, $M_u = 1/r$, and $K_u = M_u - 1$. Replacing these values in equations (4) and (5) we get,

$$\frac{C'_t}{C_t} = \frac{3r + 4\alpha - 4\alpha r}{2(1+r)} \tag{8}$$

$$\frac{C'_s}{C_s} = \frac{1}{2r}(r + 2\alpha(1-r)) \tag{9}$$

4.2 Performance Evaluation

The performance of the heuristics depends on (i) the relative cost of setting and traversing the forwarding pointers, and, (ii) call and mobility patterns of the user.

Using forwarding pointers in addition to HLS updates make sense for users with high mobility. Instead of updating the HLS on every move, forwarding pointers are used. Thus, the

effectiveness of heuristics are to be checked for call-mobility ratios, $r \leq 1$. A good heuristic should have both the relative total and search cost less than 1.

4.2.1 Performance of Movement-based Heuristic

It is observed in figures 5-7, that, the total cost increases with M and α. For high α values, the movement-based heuristic performs poorly compared to the basic HLS scheme. From Figure 8, it can be observed that the relative search cost increases with M and α. For good performance for a wide range of αs, M should be less than 4. Considering total cost as the performance index, movement-based heuristic performed better than the basic HLS scheme for the following call-mobility ratios :

$$r \leq \frac{2}{M} \frac{(M-1)(1-\alpha)}{(2\alpha(M-1)-1)} \tag{10}$$

This was obtained from equation(6).

4.2.2 Performance of Call-based Heuristic

The variation of the relative total cost with call-mobility ratio for call-based heuristic is sensitive to α. For low α, the relative total cost increases with r. This is because, as r increases, the relative update cost increases, and the search cost decreases. However, the decrease in search cost is less compared to the increase in the update cost. And since, the total cost includes the search cost and the update cost, there is an increase in the total cost as r increases.

However, for high α, the relative cost decreases with r. This is because for high values of α, as r increases, the decrease in the search cost is much more than the increase in the update cost. The performance of call-based heuristic is poor for low values of r. This is due to very high search costs.

Figure 9 illustrates the variation of relative search costs with r for various values of α. It can be observed that the search cost is very high for low values of r. This is because the location updates occur very infrequently, and the moves more frequent. Thus, each search has to traverse a long chain of forwarding pointers. Moreover, there is a rapid decrease in search costs as r increases. This is because the length of chain of forwarding pointers reduces as calls arrive more often. We observe from Figure 8 and Figure 9 that the search costs of call-based heuristic are much lower than movement-based heuristic for $r > 0.3$. Considering total cost as the performance index, call-based heuristic performed better than the basic HLS scheme for the following call-mobility values :

- low α : $r \leq \frac{|2(1-2\alpha)|}{|(1-4\alpha)|}$
- high α : $r \geq \frac{|2(1-2\alpha)|}{|1-4\alpha)|}$

This was obtained from equation(8).

4.2.3 Call-based vs. Movement-based

Now that we have determined the call-mobility ratios for various α, for which the heuristics perform better than HLS scheme, we now determine when call-based is better than movement-based heuristic. This can be found from equations (6) and (8). If there is a movement-based heuristic scheme which updates upon M moves, the call-based heuristic scheme will perform better for the following call-mobility values :

- $\alpha \leq \frac{1}{(1+M)}$ and $r \leq \frac{1}{M}$.
- $\alpha > \frac{1}{(1+M)}$ and $r \geq \frac{1}{M}$.

4.2.4 Observation

Forwarding pointers are not beneficial in environments where r is very low and α is very high. The following table gives us a fair idea about when forwarding pointers are beneficial. If they are beneficial, the table shows which heuristic performs the best.

r \ α	LOW < 0.5	AVG. 0.5	HIGH > 0.5
LOW < 0.5	Call -based	Movement -based	Not Beneficial
AVG. 0.5-1	Movement -based	Call -based	Call -based
HIGH > 1	Movement -based	Call -based	Call -based

Figure 4: Performance Chart

Movement-based updates might cause unnecessary updates even if the host does not move far since it moves in and out of a registration area frequently (thereby causing a lot registration area crossings). On the other hand, when the call-mobility ratio is very high, call-based updates will cause a lot of unnecessary updates. However, if the cost of forwarding pointers is relatively high as compared to cost of updates, call-based updates are preferable even for high values of call-mobility ratios.

5 Using Search Updates

The search costs for call-based heuristic is very high for very low values of r. This is because, the location updates occur very infrequently, and the moves more frequent. Thus, each search has to traverse a long chain of forwarding pointers. The search costs of movement-based heuristic increases linearly with M. This is again due to the fact that a search, on an average, has to traverse a longer chain of forwarding pointers.

The search cost could be reduced by eliminating the cost of querying the HLS for the destination location server (starting point of the chain). This can be achieved by having a location entry at the src_ls pointing to the starting of the chain.

A search-update occurs after a successful *search*, when the requesting host updates the location information corresponding to the mobile host. Search updates might reduce the search cost (call set-up time). The use of search updates have previously been discussed for network architectures without home location servers [1].

- *No update* : There are no search-updates. This is the case that has been analyzed till now.
- *Jump update* : A location entry is created at the src_ls for *dest*. As stated earlier, location update at the HLS take place only after a search for call-based schemes. For *jump update*, in addition to updating the HLS, the location information at src_ls is also updated. If there are no location entries for *dest* at src_ls, a location entry is created. The cost of *jump update* is the cost of creating a location entry (e.g., a forwarding pointer). The motivation behind this kind of update is based on the assumption that *src* communicates frequently with *dest*. Therefore, the subsequent search by *src* for *dest* will be lower[3]. This approach is similar to the approach of caching location data [12].
- *Path Compression update* : The forwarding pointers in the chain of location servers are updated to point to $curr\_ls$. A location entry (e.g., a forwarding pointer) is also created at src_ls. This kind of update is to be used for a highly searched host. Therefore, there is a possibility that search for *dest* is initiated from one of the registration areas in the chain. If so, the search cost will be reduced. In *jump update*, if *src* moves out of src_ls after the search update of *dest* at src_ls, the search update will not be beneficial in reducing the cost of searching *dest*. However, in *path compression update*, there is still a possibility that *src* moves into one of the registration areas which has the current location information of *dest* due to *path compression update*. Therefore, the subsequent searches for *dest* will still incur low search costs.

[3]This is true if *src* has not moved out of the registration area of src_ls.

Path compression update could be done during the search process itself. This is possible, if the forwarding pointers are bi-directional, i.e., there are backward pointers in addition to the forwarding pointers. Therefore, after a search, location of *dest* could be passed via these backward pointers to src_ls. The updates at the location servers on the chain could be done during this backward traversal. The cost of sending the location information of *dest* to src_ls using the backward pointers will be kF, which is also the conservative estimate of cost($curr\_ls \rightarrow src\_ls$) (as explained in the previous section). Therefore, no additional cost will be incurred due to the updates of the forwarding pointers in the chain. Therefore, the cost of *path compression update* will be equal to the cost of *jump update*, i.e., the cost of creating a location entry at src_ls.

5.1 Altered Search Procedure

If *dest* is not in the registration area of src_ls, check if src_ls has a location entry for *dest*. If so, follow the chain of forwarding pointers to get the current location of *dest*. Else, follow the previously explained search procedure (section 3.2.2).

5.2 Purging of Location Information

In call-based scheme a forwarding pointer is used only once. Once a chain is traversed, the same chain is not going be used again. This is because the HLS gets updated after every call. Therefore, a forwarding pointer could be purged as soon as it is used once. But the same cannot be done if search-updates are used, because, the location entry for *dest* at src_ls might point to a location server which could have purged its forwarding pointer as some other host used the chain of forwarding pointers to locate *dest*.

The location servers maintains a time associated with each forwarding pointer. It also maintains time associated with the location entries created due to search-updates. While creating (updating) a forwarding pointer/location entry, the time is the current time at the location server during creating (updating). Every location server purges the forwarding pointers/location entries which are older than T_{purge} units of time. Through this process, we avoid maintaining any stale forwarding pointers/location entries at the location servers. In addition to this, we have to require the mobile host to update its location information at its HLS at least once every T_{purge} units of time. This will ensure that even if the forwarding pointers are purged off, the HLS has the current location information of the host. This is also applicable in cases where the host does not move (movement-based) or get a call (call-based) for T_{purge} units of time. T_{purge} will be a system design parameter. Higher the value of T_{purge}, lower will be volume of updates due to purging.

5.3 Performance Analysis

- s = probability that a forwarding pointer to the destination host exists at the location server of the caller's registration area.
- $C''_u, C''_s, C''_{su}, C''_t$ are the corresponding update, search, search-update and total cost using HLS and forwarding pointers. $C''_t = C''_u + r(C''_s + C''_{su})$.

In the analysis here, we assume that T_{purge} is very large (effectively ∞). Therefore, we assume that updates at the HLS take place only upon searches (with call-based heuristic) or upon moves (with movement-based heuristic). We analyze the performance improvement of call-based and movement-based heuristics due to search-updates. As stated earlier, the cost of search-update, C''_{su}, is basically the cost of creating a location entry at src_s. C''_{su} is assumed to be equal to F. The update cost C''_u is same as C'_u.

$$C''_s = X + (1-s)(cost(HLS \rightarrow src\_s)) + 2kF$$

Assuming that the cost of HLS interactions dominates over X, we get,

$$C''_t = C'_u + r((1-s)C_s/2 + (2k+1)F)$$

Replacing $C_u = C_s$, and, $F = \alpha C_u$, we get the following expressions for the relative total cost, and relative search cost :

$$\frac{C''_t}{C'_t} = 1 - \frac{r(s-2\alpha)}{2C'_t} \tag{11}$$

$$\frac{C''_s}{C'_s} = 1 - \frac{r(s-2\alpha)}{2C'_s} \tag{12}$$

For high values of s ($s > 2\alpha$), $s \leq 1$, i.e., for low α values, and, for high probability of a forwarding pointer to the destination host existing at the location server of the caller's registration area, it is beneficial using search-updates. As shown in the following table and figures 10-12, the search cost and the total cost using search-update is lower. Table 1 illustrates the effect of s on the relative search cost for movement-based updates. Figure 10 illustrates the reduction in the total cost for movement-based updates using search-updates. As s increases, the savings in total cost increases. As shown in figures 11 and 12, similar performances are observed for call-based updates.

6 Future Work

• **Detailed Analysis** : Till now, we have analyzed the location management schemes for call-mobility patterns which have all calls originating from outside the current registration area, and,

Table 1: Relative Search Cost for Movement-based : $M = 3$, $\alpha = 0.1$

s	Relative Search Cost
0.7	0.722
0.9	0.611

all moves that are registration area crossings. However, in reality, the calls are a mixture of long-distance and local calls. Likewise, the moves are a mixture of moves within the same registration area and registration area crossings. We are currently analyzing the schemes for such call-mobility patterns.

• Fault Tolerance : The use of forwarding pointers introduces other problems. The failure of a location server containing a forwarding pointer, causes the host to be intractable. The same could also be said about the basic HLS scheme. The failure of the HLS causes the host to be intractable. This paper assumes that the failure rate of location servers are very low. Under this assumption, the modified scheme should perform correctly most of the time. Handling location server failures is an independent problem that we are studying at present.

7 Conclusions

In this paper we presented location management strategies using forwarding pointers and *search-updates* for network architectures with HLSs. We also presented two heuristics to limit the length of the chain of forwarding pointers. We analyzed the performance of these heuristics, and compared them with the scheme used in the IS-41 standard. We observed that using forwarding pointers was beneficial for most of the call-mobility ratios. However, when the relative cost of forwarding pointer operations are high, forwarding pointers are not beneficial for low call-mobility ratios. We also observed that the call-based heuristic performed better than movement-based heuristic for most of the call-mobility ratios and α values. Using call-based heuristic reduces the call set-up time (search cost) significantly for medium and high values of call-mobility ratios and α.

References

[1] P. Krishna, N. H. Vaidya and D. K. Pradhan, "Location Management in Distributed Mobile Environments," *Proc. of the Third Intl. Conf. on Parallel and Distributed Information Systems,* pp. 81-89, Sep. 1994.

[2] T. Imielinski and B. R. Badrinath, "Wireless Computing: Challenges in Data Management," *Communications of ACM,* pp. 19-28, Oct. 1994.

[3] Kathleen S. Meier-Hellstern, et. al., "The Use of SS7 and GSM to support high density personal communications," *Proc. of the Winlab workshop on third generation wireless information networks,* April 1992.

[4] L. J. Ng, et. al., "Distributed architectures and databases for intelligent personal communication networks," *Proc. of IEEE ICWC,* 1992.

[5] R. J. Fowler, "The Complexity of using Forwarding Address for Decentralized Object Finding," *Proc. ACM SIGCOMM Symposium on Communication, Architectures and Protocols,* 1986.

[6] B. R. Badrinath, T. Imielinski and A. Virmani, "Locating Strategies for Personal Communication Networks," *Proc. of the IEEE GLOBECOM Workshop on networking of Personal Communication,* December 1992.

[7] R. J. Fowler, "The Complexity of using Forwarding Address for Decentralized Object Finding," *Proc. ACM SIGCOMM Symposium on Communication, Architectures and Protocols,* 1986.

[8] Amotz Bar-Noy and H. Kessler, "Tracking Mobile Users in Wireless Communication Networks," *Proc. Infocom*, 1993.

[9] W. C. Y. Lee, *Mobile Cellular Communications Systems,* McGraw Hill, 1989.

[10] D. M. Balston and R. C. V. Macario, *Cellular Radio Systems,* Artech House, 1994.

[11] S. Mohan and R. Jain, "Two User Location Strategies for Personal Communication Services," *IEEE Personal Communications,* Vol. 1, No. 1, 1994.

[12] S. F. Wu and Charles Perkins, "Caching Location Data in Mobile Networking," *IEEE Workshop on Advances in Parallel and Distributed Systems,* October 1993.

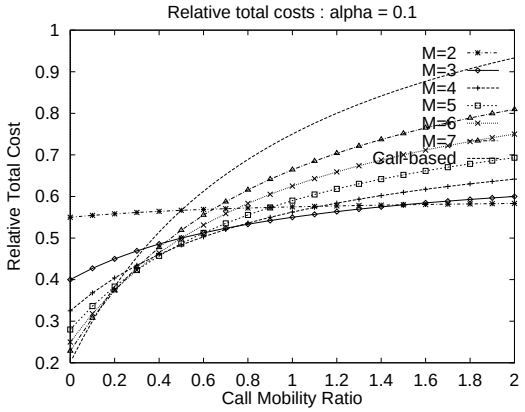

Figure 5: Relative Total Costs : $\alpha = 0.1$

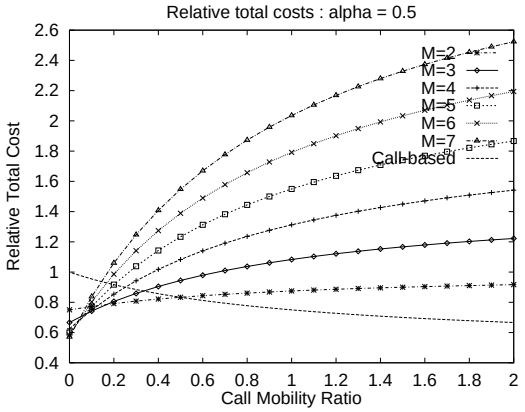

Figure 6: Relative Total Costs : $\alpha = 0.5$

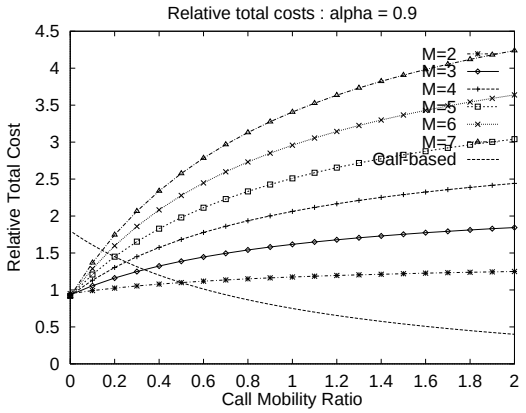

Figure 7: Relative Total Costs : $\alpha = 0.9$

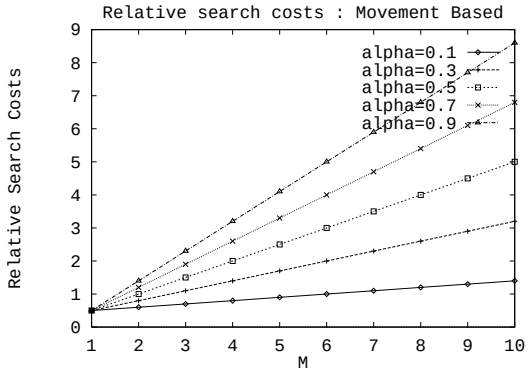

Figure 8: Relative Search Costs : Movement-based

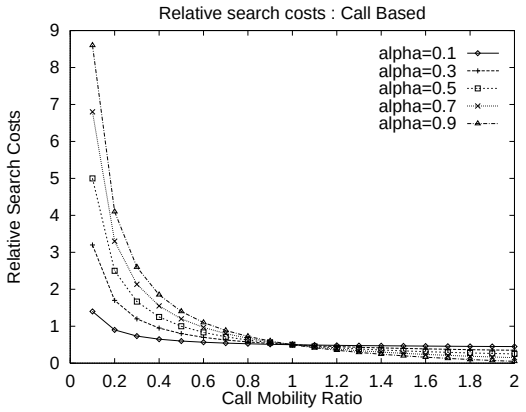

Figure 9: Relative Search Costs : Call-based

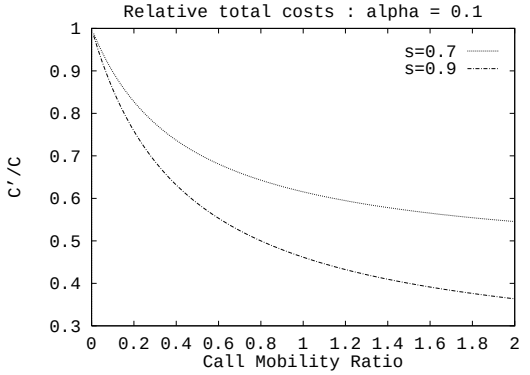

Figure 10: Relative Total Cost for Movement-based : $M = 3, \alpha = 0.1$

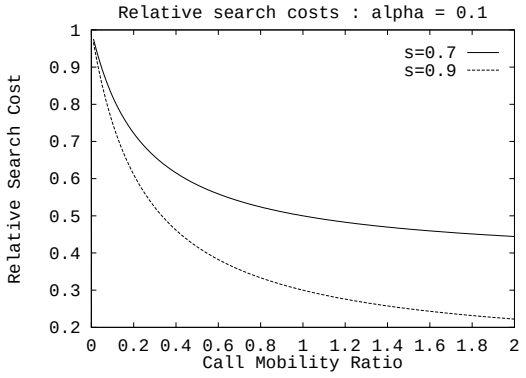

Figure 11: Relative Search Cost for Call-based : $\alpha = 0.1$

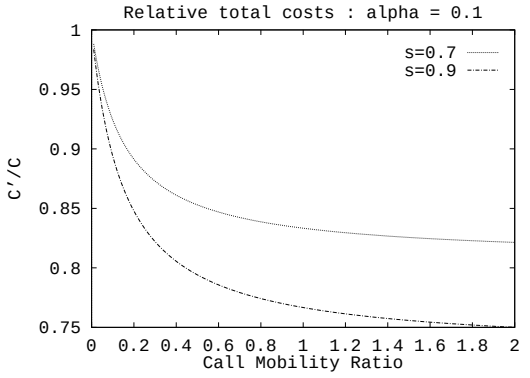

Figure 12: Relative Total Cost for Call-based : $\alpha = 0.1$