Static and Dynamic Location Management in Distributed Mobile Environments *

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
Phone: (409) 862-2599
June, 1994

Technical Report # 94-030

Abstract

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 mobile host. Various strategies can be designed for search, update and search-update. *Static* location management uses one combination of *search*, *update* and *search-update* strategies throughout the execution. Simulations were carried out to evaluate the performance of different static location management strategies for various communication and mobility patterns. It was noticed that performing *search-updates* significantly reduced the search costs with very little cost to pay for updates (upon moves and searches).

In order to obtain good performance using static location management, the system designer should a priori have a fair idea of the communication and the mobility pattern of the users. Having this information, the system designer can select the combination which performs best for the given values of communication and mobility. The host behavior (communication frequency, mobility) is not always available to the system designer. Thus, there is a need for *dynamic* location management. In this paper we present a scheme for *dynamic* location management. The basic philosophy behind dynamic management is that the past history of the system will reflect the behavior in the future. Hence, by keeping track of the past history and modifying the management strategy accordingly, one expects to perform well for any call and mobility pattern. Simulation results show that the performance of *dynamic* location management is better than static location management.

*Preliminary version will appear in the Proceedings of Third International Conference on Parallel and Distributed Information Systems, 1994. Research reported is supported in part by AFOSR
†Direct all correspondence to P. Krishna.

1 Introduction

Mobile computing is rapidly becoming a major trend in the communications market. Mobile computing gives users the information accessing capability regardless of the location of the user. Users of portable computers would like to carry their laptops with them whenever they move from one place to another and yet maintain transparent network access through the wireless link. With the availability of wireless interface cards, mobile users are no longer required to remain confined within the static network premises to get network access. In order to communicate with an user, one needs to know the user's location. Thus, the network faces a problem of continuously keeping track of the location of each and every user. This problem becomes noticeable when the network sizes are large.

Location management is one of the most important problem 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 mobile host. The goal of a good location management scheme should be to provide *efficient* searches and updates. The cost of a location update and search is characterized by the number of messages sent, size of messages and the distance the messages need to travel. An efficient location management strategy should attempt to minimize all of these parameters.

Providing connection-oriented services[5, 6, 7, 8, 12] to the mobile hosts requires that the host be always connected to the rest of the network in such a manner that its movements are transparent to the users. This would require efficient location management in order to minimize the time taken for updates and searches, so that there is no loss of connection.

In this paper we present several location management strategies based on a hierarchical tree structure database. These strategies try to satisfy the goal of providing efficient searches and updates. A location management strategy is a combination of a *search* strategy, an *update* strategy, and a *search-update* strategy. *Static* location management uses one combination of *search*, *update* and *search-update* strategies throughout the execution. This paper presents the results of simulations carried out to evaluate the performance of various static location management strategies for various call and mobility patterns.

In order to obtain a good performance using static location management, the system designer should a priori have a fair idea of the call and the mobility pattern of the users. Having this information, the system designer can select the combination which performs best for the given values of call and mobility. This information is not always obtainable. Thus, there is a need for *dynamic* location management. The basic philosophy behind dynamic management is that the past history of the system will reflect the behavior in the future, and hence by keeping track of the

past history and modifying the management strategy accordingly, one expects to perform well for any call and mobility pattern. In this paper we present preliminary ideas and results for dynamic location management.

This paper is organized as follows. Section 2 presents a review of related literature. Section 3 presents the system model for a distributed system with mobile hosts. Section 4 presents the static location management strategies, and Section 5 presents the simulation results for the various static location management strategies. Section 6 presents the dynamic location management scheme and conclusions are presented in Section 7.

2 Review of Related Literature

Numerous location strategies have been proposed in the recent years. One of the earlier works which dealt with object tracking was done in 1986 by Fowler [10]. Fowler deals with techniques to efficiently use forwarding addresses for finding decentralized objects. The environment is an object-oriented computer system, where the objects are allowed to move between processes. If a process wants to perform an operation on an object, it has to first locate it. Fowler proposed the use of forwarding pointers to keep track of these objects. As explained later, our paper borrows the idea of manipulating forwarding pointers upon a successful search.

Awerbuch et. al. proposed a theoretical model for online tracking of mobile hosts [3]. The architecture is assumed to be a hierarchy of "m-regional matching directories". Each node u in the data-structure maintains sets $read(u)$ and $write(u)$. The sets are such that, if there is a node v which is at a distance of less than m from u, the intersection of $read(u)$ and $write(v)$ is non-null. The same applies for $read(v)$ and $write(u)$. The cost of moves and updates was derived to be polylograthmic in the size and diameter of the network. They also use forwarding pointers. Regional matching directories are used to enable localized updates and searches.

Badrinath et. al. examined strategies that reduced search costs and at the same time control the volume of location updates by employing user profiles [2]. The architecture consists of a hierarchy L of location servers which are connected to themselves and to the base stations (or mobile support stations) by a static network. Each user is assumed to be registered under one of the location servers called the *home location server* (HLS). The user profiles were used to create partitions. When the user crosses partitions, does the update takes place. As explained later, our paper uses a similar architecture, but, does not assume a *home location server* (HLS).

Spreitzer et. al. proposed a network architecture which consists of *user agents*, and a *location query service* (LQS) [14]. The paper deals with tracking of hosts and providing some privacy to the user. The user agents are responsible to forward any communication to or from the user. There is a dedicated user agent per user. Currently the location tracking is done using mechanisms such as

infra-red-based active badge tracking, device input activity from various computers, and explicitly specified information obtained directly from the users. The user agent always know the current location of the user. The *LQS* is used to provide ways of executing different location queries that offer different trade-offs between efficiency and privacy. This scheme was mainly aimed for local networks such as in building premises. As the number of hosts in a network increase, it might not be efficient to have a dedicated user agent per user.

Wu et. al. dealt with the idea of caching location data at the Internet Access Point(*IAP*) [11]. Here, the *IAP* will maintain location data of some of the hosts. This becomes useful when optimal routing decisions are to be taken. If the *IAP* did not have an entry for a host, the message is forwarded to the *Mobile Router* (*MR*) which maintains information of all the hosts. It is a very simple idea that will be effective for local networks. However, when the network sizes increase, the *MR* will become a serious bottleneck, and one has to resort to more efficient location management techniques.

3 System Model

In this section we present a model for a distributed system with mobile hosts. As shown in Figure 1, mobile networks generally comprise of a static backbone network and a wireless network. There are two distinct sets of entities, namely mobile hosts and fixed hosts. A host that can move while retaining its network connection is called a *mobile host* (*mh*) [1]. The static network comprises of the fixed hosts and the communication links between them. Some of the fixed hosts, called *mobile support stations* (*MSS*)[1] 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*. 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 [1]. As the demand for services increase, the number of cells may become insufficient to provide the required grade of service. *Cell splitting* [15] 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 [4]. 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. Due to mobility, *mh* may cross the boundary between two cells while being active. Thus, the task of forwarding data between the static 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 [4]. The initiative for a handoff can come

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

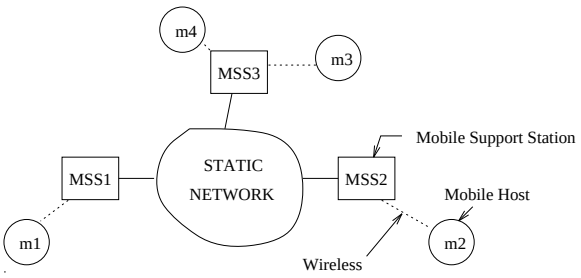


Figure 1: Model of a distributed system with mobile hosts

from the mobile host or the mobile support stations. Handoff helps to maintain an end-to-end connectivity in the dynamically reconfigured network topology.

4 Static Location Management

A location management strategy will be a combination of the *search* strategy, an *update* strategy, and a *search-update* strategy. Figure 2 illustrates the space of location management strategies discussed in this paper. In this paper, we are going to discuss location management strategies in the absence of a *home location server* (HLS).

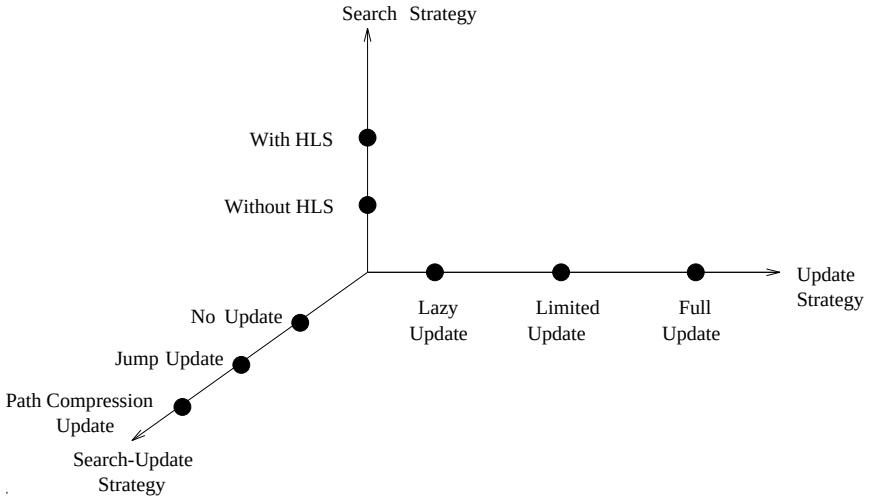


Figure 2: Space of Location Management Strategies

4.1 Logical Network Architecture (LNA)

Mobile systems consist of mobile hosts, mobile support stations (base stations), and location servers. The logical network architecture (LNA) is a hierarchical structure (tree) consisting of

mobile support stations and *location servers*[2]. As shown in Figure 3, the mobile support stations (MSS) are located at the leaf level of the tree. Each MSS maintains information of the hosts residing in its cell. The other nodes in the tree structure are called location servers (LS). Each location server maintains information regarding mobile hosts residing in the subtree beneath it.

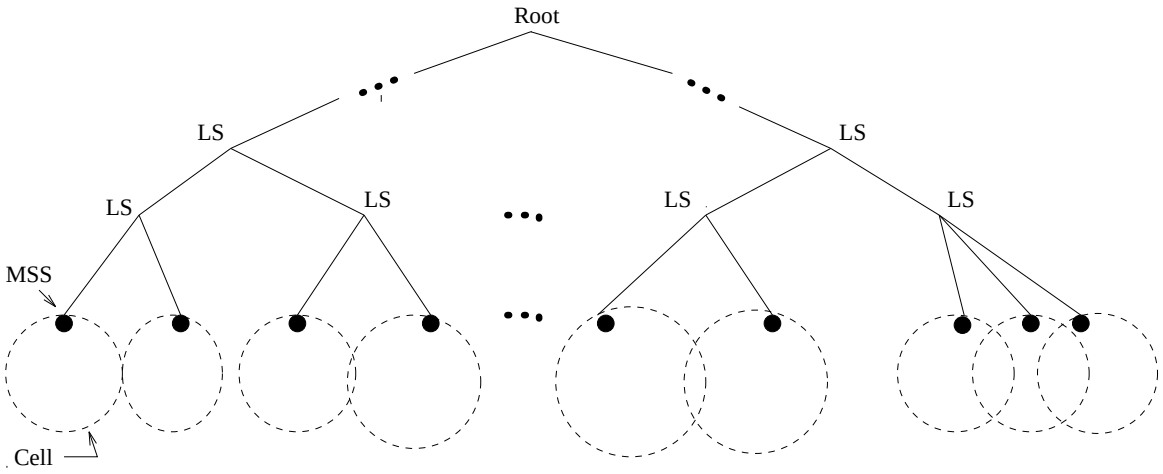


Figure 3: Logical Network Architecture

Each communication link has a weight attached to it. The weight of a link is the cost of transmitting a message on the link. Let $l[src][dest]$ represent the link between nodes src and $dest$, and, let $w(l)$ represent the weight of the link l. The cost depends on the size of the message, the distance between the hosts, and the bandwidth of the link. For analysis purposes, we assume that, for all l, $w(l) = 1$. Essentially, our cost metric is the number of messages.

4.2 Data Structures

There is an unique "home" address for every mobile host. The home address is the identifier/name of the mobile host. The "physical" addresses of a mobile host might change, but its home address remains the same, irrespective of the host's location [12, 13]. Each LS maintains an address matching table that maps the home address to the physical address of the mobile hosts residing in the subtree beneath it. Thus, the problem of location management basically focuses on the management of the address matching table.

There is a location entry in LS corresponding to a host h, if the host h is in one of the cells in the subtree of a location server LS. If the host h moves to a cell which is not in the subtree of LS, then the entry corresponding to h is updated (as explained later) at LS. All the nodes maintain location information using 3-tuples which have the following elements : (i) Mobile host

[2]Typically location servers correspond to the mobile switching centers.

identifier (id), (ii) Forwarding pointer destination (fp_dest), and, (ii) Time at which last forwarding pointer update took place (fp_time). Each location server maintains a 3-tuple for each mobile host residing in the subtree beneath it, and each mobile support station maintains a 3-tuple for each mobile host residing in its cell.

The default value of fp_dest and fp_time is $NULL$. Forwarding pointer destination (fp_dest) is the location of mobile host. If the fp_dest field of a host h is $NULL$ in the location server L, then, h is not in one of the cells in the subtree of L. Let us illustrate the use of forwarding pointers with an example. Let us suppose that we are using a strategy which uses forwarding pointers for location updates. Let a host h reside initially in cell c. The MSS of the cell c will have an entry $(h, NULL, NULL)$. Let there be a location server L which maintains information of the hosts residing in cell c. There will be an entry $(h, c, NULL)$ corresponding to host h at L. Let host h move to a new cell c', which is not a part of the subtree of L. Let t be the local time at the MSS of cell c at which the change of location of h is recorded at the MSS. Let t' be the local time at L at which the change of location of h is recorded at L. Thus, the location information of h will be (h, c', t') at L, and, (h, c', t) at MSS of cell c.

A Note: The above data structures contain fp_time field to store time. The fp_time entry for a data structure on a node, say v, contains the local time at node v when the data structure was last modified. We will denote this time as t in the following. It should be noted that the correctness of the algorithms does not require the clocks at various nodes to be tightly synchronized.

4.3 Initial Conditions

It is assumed that, initially, the location information of the mobile hosts is stored in the corresponding location servers i.e., each location server (LS) should be having the correct location information for all the hosts residing in the cells in its subtree. Thus, the root location server should have the correct location information of all the hosts in the system. Let us illustrate this with an example. In Figure 4, nodes 1-7 are location servers, and 8-15 are mobile support stations. There are two mobile hosts $h1$ and $h2$. In the initial state, host $h1$ is in cell 8, and $h2$ is in cell 12. Initially, the correct location information of host $h1$ will be available at the location servers {4,2,1}. Likewise, the location information of $h2$ will be available at the location servers {6,3,1}. Thus, the location information of a host is available at all the location servers located on the path from its current MSS to the root.

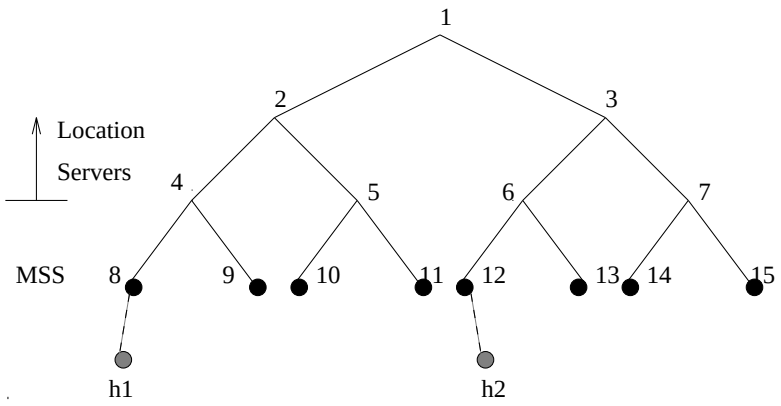


Figure 4: An Example

4.4 Update Protocols

The strategies for updating the location information at the location servers and the mobile support stations, when the host moves[3], are as follows.

Let *src* and *dest* be the identifier of the source and destination cell, respectively. Let h be the identifier of the mobile host. Let t be the local time at the node at which the change of location of h is recorded at the node. i.e., for example, t is the local time at the MSS of *dest* at which a location entry is added at the MSS.

4.4.1 Lazy Updates (LU)

This is the simplest update scheme. Updates take place only at the MSS of the source and destination cells. A forwarding pointer is kept at the source MSS. The updated entry at the source MSS becomes $(h, dest, t)$. An entry for host h, $(h, NULL, NULL)$ is added at the destination MSS. The location information at the location servers are not updated. The cost of update is zero, because there are no update messages being sent.

4.4.2 Full Updates (FU)

Upon a move, apart from the MSSs involved (i.e., the MSS of the source and destination cells), location updates take place in all the LSs located on the path from the MSS of the source and destination cells to the root. The scheme and an example illustrating it follows.

Source cell:

[3]Here movement of the host implies that the host crosses cell boundary.

1. At the MSS : For host h, set $fp\_dest = dest$, and $fp\_time = t$. The updated entry for host h at the MSS becomes $(h, dest, t)$.

2. All location servers on the path from src to the root : The MSS of src sends update message to these location servers. Upon receipt of the update message, the location servers update the entry for h to $(h, dest, t)$.

Destination cell:

1. At the MSS : An entry $(h, NULL, NULL)$ is added for host h. If there was an old entry for h, it is overwritten by this new entry. There can be only one entry per host (i.e., MSS or location server).

2. All location servers on the path from $dest$ to the root : The MSS of $dest$ sends update message to these location servers. Upon receipt of the update message, the location servers create an entry $(h, dest, NULL)$. If there was an old entry, it is overwritten by this new entry.

Therefore, in an H-level tree, the update cost[4] per move is $2(H-1)$. Let us illustrate this scheme with an example. Suppose in Figure 4, host $h1$ moves from 8 to 14. Forwarding pointer to 14 will be kept at MSS 8. MSS 8 sends update message to {4,2,1}, and these location servers also maintain forwarding pointer to 14. An entry for $h1$ will be made at MSS 14. MSS 14 sends update message to the location servers {7,3,1}, and these location servers also make an entry for host $h1$.

4.4.3 Limited Updates (LMU)

Update in the location information takes place at a limited number of level of location servers in the tree. Here updates occur at $m(< H)$ lower levels of location servers on the path to the root. Updates at these location servers are similar to the FU scheme. The location servers at levels higher than m are not updated. Thus, the update cost per move is $2m$. Let us illustrate this scheme with an example. Let the value of m be chosen to be 1. Suppose in Figure 4, host $h1$ moves from 8 to 14. Forwarding pointer to 14 will be kept at MSS 8. MSS 8 sends an update message to {4}, and 4 maintains forwarding pointer to 14. An entry for $h1$ will be made at MSS 14. MSS 14 sends an update message to {7}, and 7 makes an entry for host $h1$.

[4] As stated earlier, the cost metric is the number of messages.

4.5 Search Protocol

If a host h in cell C wants to communicate with another host h', h has to know the location of h'. This requires that host h search for host h'. As stated earlier, we do not make explicit use of *home location server* (HLS) for searches. The search process in the absence of a HLS is as follows. If the mobile support station of C has no location information for h', it forwards the location query to the next higher level location server on the path to the root. If that location server does not have any location information for h', it again forwards the location query to the next higher level location server on the path to the root. This process repeats until a location server which has location information for h' is reached. Once the location information (cell identifier) for h' is obtained, the location query is forwarded to the MSS of the cell. Host h' is either in the cell of MSS, or, MSS has a forwarding pointer corresponding to h'. If host h' is in the cell of MSS, the search is complete. Else, a chain of forwarding pointers is traversed till the MSS containing the host h' is reached. The search protocol is as shown in Figure 5.

Initially, the *search* cell is C.
Step 1 : **If** the MSS of the *search* cell has an entry for h',
If $fp\_dest = NULL$, host h' is in the *search* cell. Search for h' is complete.
Else *search* cell = fp_dest. Repeat step 1.
Else forward the query to the next higher level location server on the path to the root.
Step 2 : **If** the location server has an entry (h', fp_dest, fp_time) for h' :
If $fp\_dest \neq NULL$, *search* cell = fp_dest. Go to step 1.
Forward the query to the next higher level location server on the path to the root.
Go to Step 2.

Figure 5: Search Protocol

4.6 Search-Update Protocols

Location management becomes more efficient if the location updates also take place after a successful search. For example, suppose there is a host h that frequently calls h', and h' is highly mobile. It makes sense to update the location information of h' after a successful search, so that in the future if h calls again, the search cost is likely to reduce. The location information update takes place at the MSS of the caller. Let host h be the caller, and host h' be the destination host. Let the location of h and h' be C and C' respectively. Following are the strategies to update location information upon a search.

4.6.1 No Update (NU)

In this strategy, there are no location updates. But, the fp_time field of the entry corresponding to h' at the MSSs on the search path are updated to the current time at the MSS. The cost is zero. This is because the update of the time field could be done during the search process itself, and no additional messages need to be sent for this purpose. The update in fp_time is done to avoid *purging* of the forwarding pointer data at the MSSs. The *purge* protocol is explained in the next section.

4.6.2 Jump Update (JU)

In this strategy, a location update takes place only at the caller's MSS, i.e., MSS of the cell C. The entry for h' at the MSS of cell C is set to (h', C', t), where t is the local time at the MSS when the location information is updated. The update cost is 1. This is because only one message needs to be sent from the MSS of C' notifying the location information of host h'.

4.6.3 Path Compression Update (PCU)

In this strategy, upon a successful search, a location update takes place at all the nodes in the search path. All the location servers on the search path have the entry of h' updated to (h', C', t), where t is the local time at the location server when the location information is updated. All the MSSs on the search path including the caller's MSS have an entry of h' updated to (h', C', t), where t is the local time at the MSS when the location information is updated. Let us illustrate with an example. In Figure 4, let host $h1$ call host $h2$. Suppose the location information of $h2$ is available only at the location servers {6,3,1}. Using the search protocol described previously, the search path will be $8 \rightarrow 4 \rightarrow 2 \rightarrow 1 \rightarrow 12$. The location updates take place at location servers {4,2,1}, and MSS 8. The update cost is the length of the search path, which in this example is 4.

4.7 Purging of Forwarding Pointers

We need to periodically purge the stale forwarding pointers at the location servers and the mobile support stations. This should be done in order to (i) save storage space at the nodes, and (ii) avoid storing stale location information. It does not make any sense to keep the forwarding pointer information for a host h, if no other host is going to query this location server for the location information of h. We use a design parameter *maximum threshold call interval* ($MTCI$) to decide whether to purge a forwarding pointer information or not. Let the current time be $curr\_time$. If

$fp\_time \neq NULL$, and $curr\_time - fp\_time \geq MTCI$[5], then the entry for the host is purged from the MSS. If $curr\_time - fp\_time < MTCI$, it means that there is some other host in the system which has recently used the forwarding pointer information of i.

In the location servers, if the $fp\_time \neq NULL$ and $curr\_time - fp\_time \geq MTCI$ for a host, the location entry for the host is purged.

4.7.1 Updating of Forwarding Pointers with a Purge

When LU and LMU strategies are used, the forwarding pointers at higher level location servers do not get updated, and become stale. Thus, these forwarding pointers get purged periodically. However, some of the searches for the host might reach the higher levels. If the location servers at the higher levels do not have the information of the host, the root has to broadcast to find out the location. To avoid this, the forwarding pointers at the location servers on the path to the root from the current MSS must be updated periodically along with purging. This is achieved by the current MSS of each mobile host by sending a location update message to the location servers on the path to the root.

5 Simulations

A trade-off exists between the cost of updates (upon moves and searches) and cost of searches. The parameters that affect this trade-off are (i) call frequency, and (ii) mobility. In this paper we will evaluate the effects of mobility and call frequency on the cost of updates, search-updates and searches. As stated earlier, the location management strategy is a combination of a *search strategy*, an *update strategy* and a *search-update strategy*. The search protocol is the same for all location management strategies. A total of 9 *static* location strategies are obtained using above strategies for *updates* and *search-updates*. We performed simulations to analyze the performance of the proposed location management strategies for various call frequency and mobility values. The location management strategies simulated were obtained by choosing one update strategy (say XX, where XX = LU, FU or LMU) and one search-update strategy (say YY, where YY = NU, JU or PCU). The location management strategy thus obtained is denoted as XX-YY.

5.1 Model

We assume a binary tree as the logical network architecture for the simulations. The height of the tree is H. The number of location servers in the network is $2^{(H-1)} - 1$, and the number of mobile

[5]Note that the fp_time value for a host residing in the cell will be $NULL$. So we are considering hosts which are currently not residing in the MSS's cell and whose forwarding pointer information is stored at the MSS.

support stations (or the number of cells) is $2^{(H-1)}$. Physical proximity of the cells under the same location server is assumed. This will help in determining *short* and *long* moves. The height of the tree H was chosen to be 10 for the simulations[6]. Thus, there were 512 cells in the network.

The main aim of the paper is to develop protocols for efficient searches and updates, i.e., reduce the number of messages due to location updates, without increasing the number of messages required for searches. Since, the average message delay is going to be negligible compared to the call frequency and mobility, we assume that there are no message delays, i.e., the location updates and searches are immediate. The performance of the schemes is not going to be affected by this assumption.

Simulations were performed for two types of environments : (i) *arbitrary* moves and *arbitrary* callers, (ii) *short* moves and a *set of callers*. In type (i), the user can move to any location (cell), and, get calls from any other host in the network. This is not necessarily true in real life, but it gives a fair idea of the performance of the location management schemes in such extreme conditions. Type (ii) is the closer to real life mobile environments. Users are expected to make a lot of *short* moves to nearby destinations, and are expected to receive calls from a specific set of callers (e.g. family, business colleagues)[7].

5.1.1 Call and Mobility Distribution for Type (i)

The time between moves of a host is assumed to follow an exponential distribution with a mean M. The destination cell is chosen randomly among the 512 cells. The time between calls for a host is assumed to follow an exponential distribution with a mean C. The caller's cell is chosen randomly from among the 512 cells.

5.1.2 Call and Mobility Distribution for Type (ii)

Type (ii) consists of generating calls from a specific set of callers and short moves. One option to generate short moves is to put an upper limit on the length of the move, in terms of number of cells, and randomly vary the length of the move within the upper limit. For example, in Figure 4, if we keep an upper limit of 1, the host $h2$ will be able to move to any cell in the set {11,12,13}. But, our logical network architecture just assumes proximity of cells which are under the same location server. Thus, a move from $12 \rightarrow 11$ is not equivalent to the move from $12 \rightarrow 13$.

Instead, we varied the number of levels of location servers where location information would have been updated due to the move, if FU update strategy were to be used. The number of levels

[6] In existing networks like GSM or Internet, the height will be 3 to 4. Since a binary tree was assumed for the simulations, we needed to have higher number of levels to have a sizeable number of cells in the network. However, similar performance trends are expected for other networks.

[7] The callers are assumed to be immobile. They are either part of the static network, or, do not leave their cell.

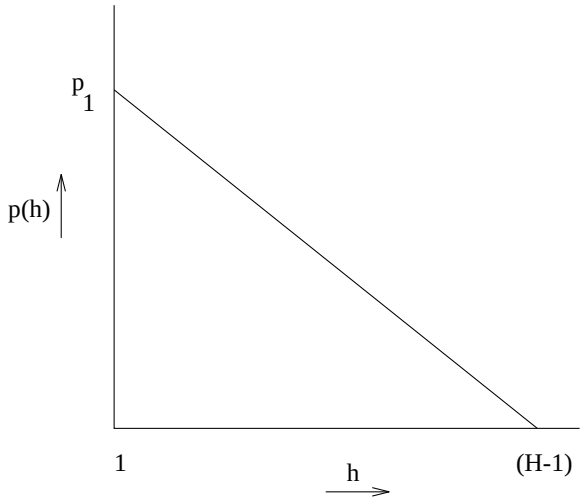


Figure 6: Probability distribution function in terms of height

can be varied between 1 to $(H-1)$. Level 0 is the MSS level. Lesser the number of levels affected, shorter is the length of the move. The probability distribution function of the length of the move in terms of height (number of levels) is shown in Figure 6.

$$p(h) = \frac{2}{(H-1)(H-2)} * (H-1-h).$$

The cumulative distribution function (cdf) is as follows: $cdf(h) = \sum_{x=1}^{h} p(x)$. We randomly chose a height h based on the given probability distribution function. The number of choices for the destination cell is 2^h. Let the identifier of the current cell (i.e., the source cell) be $curr$. Knowing the height h and $curr$, one can easily determine the ancestor of $curr$ at level h in the binary tree. Let it be ls. Knowing ls, the set of destination cells possible is $\{ls*2^h, ls*2^h+1, ..., ls*2^h+2^h\}$. A destination cell is chosen randomly from this set. Let us illustrate with an example. In Figure 4, for host $h2$, let the h obtained randomly be 1. Thus the number of choices is 2. The location server at level 1 is 6. Thus, the destination cell is randomly chosen from {12,13}. This is in coherence with the assumption of proximity of cells under the same location server. The time between moves of a host is assumed to follow an exponential distribution with a mean M.

In type (ii), for each mobile host, callers were chosen from a specific set of cells. The size of the set was chosen to be 20. The set was chosen arbitrarily, and were not necessarily neighboring cells. The calls always originated from those cells. The time between calls for a host is assumed to follow an exponential distribution with a mean C.

5.1.3 Purge

Purge is performed periodically every $MTCI$ units of time. The value of $MTCI$ was chosen to be 10 units of time.

5.2 Cost Model

As stated earlier, the cost of transmitting a message over any link is 1. Therefore, the cost metric is essentially the number of messages required for each operation (search, update, and search-update). Thus, the cost of an update is the number of location servers which update the location information of the host. The cost of a search is the number of location servers and mobile support stations visited before locating the host. Cost of a search-update is the number of location servers which update the location information of the host.

The performance parameter of interest is the aggregate cost, defined as the sum of average update cost, average search cost, and the average search-update cost.

5.3 Results

Simulations were performed to analyze the performance of the various location management strategies. Results were obtained for the two type of environments, Type (i) and (ii). The values of C and M were both varied from 1 to 15 units of time. Value of C was changed to vary the time interval between two successive calls. Value of M was changed to vary the mobility of the host. For example, $C = 1$ and $M = 1$ characterizes a communication intensive and ultra-mobile environment.

Type(i) : The average length of a move was 170, and the average distance of a call was 170 also. It was observed that the LU-PC strategy outperforms all the other strategies for all values of M and C. Therefore, we have only plotted the curves for LU-PC. The strategies using FU and LMU suffered due to the high cost of updates upon each move. LU-NU strategy suffered due to very high search costs. Because the callers were arbitrary, LU-JU strategy did not perform well as the update upon a successful search was not helping in reducing the search cost. Figure 7 demonstrates the aggregate cost for the LU-PC strategy as a function of C for different values of M. As seen in the figure, the aggregate cost increases with C, and decreases with M. This is because as C increases, the calls become infrequent, and the hosts might have moved to new locations, requiring new searches. Thus the reduction in search cost by path compression is not much effective. We also observe that the rise in aggregate cost with C is higher for lower values of M. Lower the value of M, higher is the mobility, and thus the search cost will be higher. At high values of M, the difference in the aggregate costs due to different values of M is low. This is

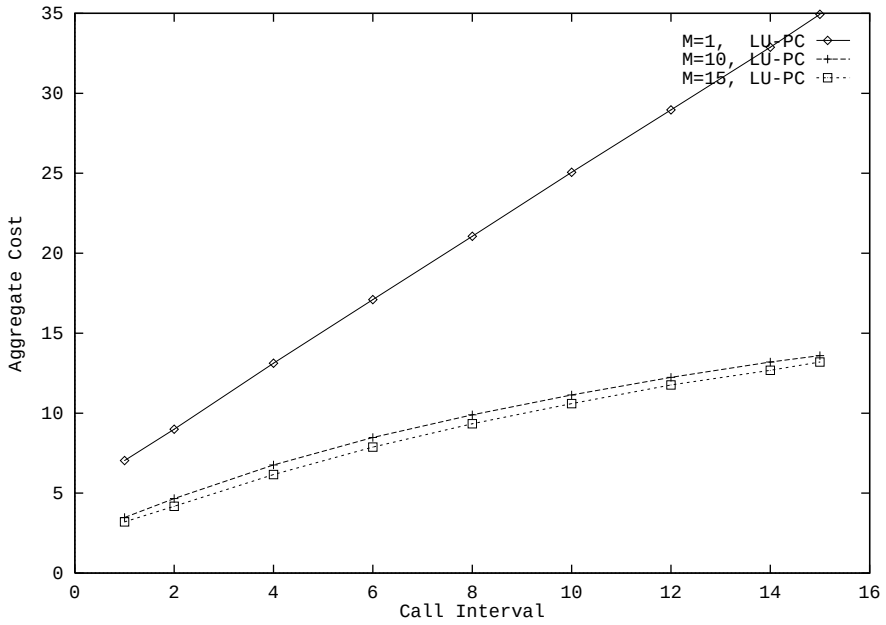


Figure 7: Performance of $LU\text{-}PC$ for type(i)

because as M increases, the host movement reduces. Beyond a point, increasing M does not affect the aggregate costs, and the curves converge to a single curve.

Type(ii) : The average length of a move was 9, and the average distance of a call was 110. It was observed that the $LU\text{-}PC$ and the $LU\text{-}JU$ strategies outperformed all the other strategies for all values of M and C. In contrast to Type (i) scenario, $LU\text{-}JU$ performed well, because, there is a specific set of callers. Thus, the jump update at the caller is much more effective in reducing the search cost, because the caller is going to call the host again with a higher probability than in Type (i) environment. Figure 8 demonstrates the aggregate cost for the $LU\text{-}JU$ strategy and the $LU\text{-}PC$ strategy as a function of C for different values of M. As seen, $LU\text{-}JU$ performs better than $LU\text{-}PC$ in high-communication and low-mobility, and, low-communication and high-mobility environments. In these environments, the search cost for $LU\text{-}PC$ and $LU\text{-}JU$ are comparable. Since the search-update cost is same as the search cost for $LU\text{-}PC$, the aggregate cost for $LU\text{-}PC$ is simply twice the search cost. Whereas, the average search-update cost for $LU\text{-}JU$ is less than[8] or equal to 1. Thus, the aggregate cost of $LU\text{-}JU$ is lower than $LU\text{-}PC$. $LU\text{-}PC$ performs better for other values of M and C because the search cost for $LU\text{-}JU$ becomes large compared to $LU\text{-}PC$. Figure 9 demonstrates the average search cost for the $LU\text{-}JU$ strategy and the $LU\text{-}PC$ strategy as a function of C for different values of M. As seen, $LU\text{-}PC$ has a much lower search cost than

[8] In cases where the caller has the correct information of the destination host, the search-update cost is zero.

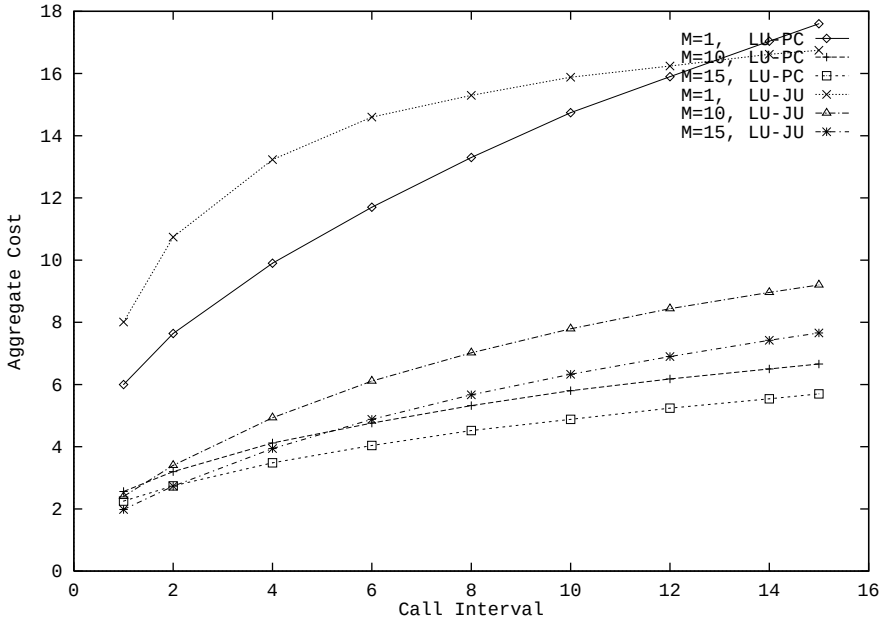


Figure 8: Comparison of *LU-PC* and *LU-JU* for type(ii)

LU-JU. The search cost of *LU-JU* is slightly lower than *LU-PC* for high-communication and low-mobility environment.

5.4 Discussion

It was noticed that performing search-updates significantly reduced the search and aggregate costs. For the logical network architecture assumed, it is seen that the *LU-PC* strategy performs better than the other strategies for most of the values of C and M. It is expected that *LU-PC* will perform well in other network models too. For models with different costs associated with each link, we expect the other proposed strategies to perform well, and sometimes better than the *LU-PC* strategy for some values of M and C. As shown in Figure 10a, we expect zones in the *M-C* plane, where one scheme will outperform others for the call frequency and mobility values in the zone. This was evident in the Type (ii) environment. As shown in Figure 10b, the *M-C* plane is divided in two zones, *LU-JU* and *LU-PC*. Thus, if the behavior of the mobile hosts (call frequency, mobility) is known a priori, the designer can obtain such an *M-C* chart and decide which location strategy will best suit the system.

In the next section we will present some preliminary ideas and results for dynamic location management.

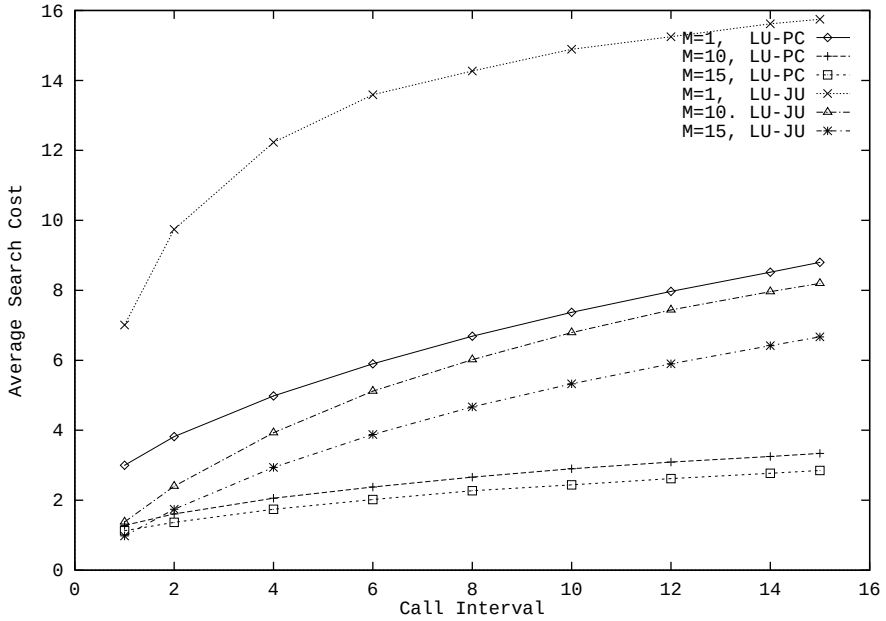
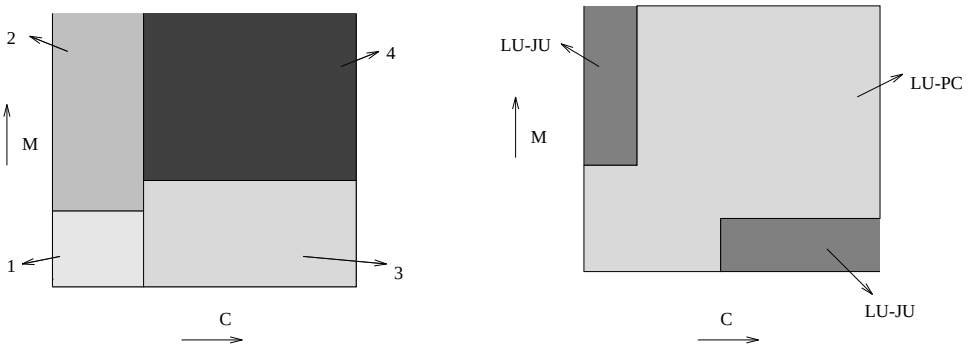


Figure 9: Comparison of search costs of LU-PC and LU-JU for type(ii)



(a) Generic Scenario (b) Type (ii) Environment

Figure 10: Partitioning of the M-C plane

6 Dynamic Location Management

The system designer does not always have prior knowledge of the mobility and the call frequency of the hosts. In these cases, one would require a location management scheme that can dynamically change the update and search-update strategy, such that the overall overhead incurred due to updates and searches is minimized. At the same time, we would not want to use up the power of the mobile hosts to determine the appropriate strategy dynamically. We require the MSS to take up the responsibility.

6.1 Data Structures

Let τ be the current time at the mobile host h. $M(h)$ is the sequence of moves of the host h. $M(h) = \{m_1, m_2, ..., m_n\}$, where, $m_1 = (t_1, src, dest)$, i.e., element m_1 is a move by the host h at time t_1 (The time of move is observed at the mobile host h.) from src to $dest$, , and $t_1 \leq t_2 ... \leq t_n$. Each element of the set $M(h)$, m_i, contains two identifiers – the source cell identifier, and the destination cell identifier. If both identifiers are the same, then the host has not left the cell. This kind of entry is not necessary (hence will not be present), because it does not affect the location database. But if the identifiers are different, the source cell should determine whether the move is *long* or *short*.

$C_u(h)$ is the sequence of costs incurred due to updates upon the moves $M(h)$. $C_u(h) = \{c_{u1}, c_{u2}, ..., c_{un}\}$, where c_{uj} = cost of update upon a move m_j.

If another host h' wants to communicate with h, and if h is not in the same cell or if the MSS of h' does not know the cell identifier of h, h' has to search for h. A set $S(h)$ is maintained at the current MSS of h. $S(h) = \{s_1, s_2, ..., s_n\}$, where $s_1 = (t_{s1}, h')$; i.e., there was a call from h' for h at time t_{s1}, and $t_{s1} \leq t_{s2} ... \leq t_{sn}$. Again, the time of call is observed at the mobile host h.

$C_s(h)$ is the sequence of costs incurred due to the searches $S(h)$. $C_s(h) = \{c_{s1}, c_{s2}, ..., c_{sn}\}$, where c_{sj} = cost of search s_j. $C_{su}(h)$ is the sequence of costs incurred due to search-updates upon searches $S(h)$. $C_{su}(h) = \{c_{su_1}, c_{su_2}, ..., c_{su_n}\}$, where c_{su_j} = cost of search-update upon the search s_j.

The data structures are obtained as explained in the next section.

6.2 Basic Idea

The above data structures are stored at the current MSS of the host. They get transferred to the new MSS during handoff. The decision of the type of updates and search-updates are done by the current MSS. The current MSS uses the data structures to determine the best suited

strategy. The appropriate update and search-update strategy will be one of the proposed static location management update and search-update strategies.

It is assumed that the mobile host h knows the identifier of the cell it is currently residing in. When a host h moves, h sends a message (containing the identifier of its old cell, and the time of move) to the new MSS. The new MSS forwards a copy of this message to the old MSS. The move is recorded as a new element m_j in the sequence $M(h)$. The old MSS takes a local decision (explained later) regarding the updates. The cost of the update is recorded as a new element c_{uj} in the sequence C_u. The new MSS requests the old MSS for the data structures corresponding to h. If the new MSS makes any updates, the cost of the update is added to c_{uj} in C_u.

When a host h' wants to communicate with h, and if h is not in the same cell or if the MSS of h' does not know the identifier of the cell of h, h' has to search for h. A location query message is sent during the search. This message has a field to store the search cost. At any time, the search cost field indicates the cost incurred due to the search till now. The search cost gets incremented as the location query message is forwarded to a new location server or a mobile support station. Once h is located, a new element s_j is added to the sequence $S(h)$ at the MSS of h. The time of the call is the time observed at the mobile host h. The search cost is recorded as a new element c_{sj} to $C_s(h)$. The MSS decides upon the appropriate search-update strategy. It is determined based on the call history (explained later). For example, if a host h' frequently calls host h, it makes sense to use JU to reduce the subsequent search cost for h'. The cost of the search-update is recorded as a new element c_{su_j} to $C_{su}(h)$ at the MSS.

6.3 Mobility and Call Frequency

6.3.1 Determining Mobility

Let at time $t = \tau$, $M(h) = \{m_1, m_2, ..., m_n\}$, where $m_n = (src, t_n, dest)$, and $t_n \leq \tau$. Thus, m_n describes the move of host h that took place at time t_n from a cell whose identifier = src to a cell whose identifier = $dest$. Let $\triangle t_i = (t_i - t_{i-1})$, where $t_0 = 0$. Thus, the average time interval between successive moves $\triangle t_{avg} = \frac{\sum_{i=1}^{n} \triangle t_i}{n}$.

We assume a system parameter *maximum threshold move interval* ($MTMI$). If there are no moves by the host for $MTMI$ amount of time, the host can be declared to be immobile or stationary. The sets $M(h)$ and $C_u(h)$ maintained at the current MSS are stale because the history does not reflect the behavior in future anymore. Therefore, they are deleted. In the absence of $M(h)$ set, the host is assumed to have a high mobility upon the first move.

We have defined two degrees of mobility – (i) low mobility, and (ii) high mobility. At any time τ, let t_n be the time of the last move by the host. If $\triangle t_{avg} < MTMI$, the host has a high mobility, else if $\triangle t_{avg} \geq MTMI$, the host has a low mobility.

6.3.2 Determining Call Frequency

Let at time $t = \tau$, $S(h) = \{s_1, s_2, ..., s_n\}$, where $s_n = (t_{sn}, h')$, and $t_{sn} \leq \tau$. Thus, s_n describes the call for host h from h' that took place at time t_{sn}. We define an average time interval between calls for **each** caller to host h. The average time interval between successive calls of caller h', $\triangle t_{savg}[h'] = \frac{\sum_{i=1}^{n'} \triangle t_{si}}{n'}$, where, n' is the number of calls made by h', and the $\triangle t_{si}$'s are the time intervals between two consecutive calls made by host h'.

We assume a system parameter *maximum threshold call interval* ($MTCI$). If there are no calls by host h' for $MTCI$ amount of time, the host h' can be declared to have no communication with h. The elements corresponding to host h' in the set $S(h)$ are stale because the history does not reflect the behavior in future anymore. Therefore, they are deleted. In the absence of $S(h)$ set, the caller h' is assumed to be a frequent caller upon the first call of h' to host h.

Similar to mobility, based on the degree of call frequency, we have two types of caller – (i) non-frequent caller, and (ii) frequent caller. Then, if $\triangle t_{savg}[h'] < MTCI$, the caller is a frequent caller, else if $\triangle t_{avg}[h'] \geq MTCI$, the caller is a non-frequent caller.

6.3.3 Size of Data Structures

The maximum size n of the move set $M(h)$ and search set $S(h)$ can be chosen as a design parameter. The storage capacity available at the MSS restricts the value of n. The MSS has to maintain these sets for each mobile host in its cell. Thus, larger the value of n, higher is the storage cost. Hence, a small value of n will be preferred. On the other hand, larger the value of n, better will be the learning of the host behavior, and thus a better predictability will be attained.

6.4 An Example

In this section we will present an example algorithm for dynamic location management. It is for the network model assumed for static location management strategies. The knowledge of Figure 10b, and the fact that $LU\text{-}PC$ is the best scheme for long moves, will prove to be useful in dynamically determining the best strategy. From the previous section, we have the techniques to classify the moves, calls and the mobility of the host. If a host has a lot of frequent callers, the host is being

frequently searched, else, if a host has a lot of non-frequent callers, the host is not frequently searched. The algorithm is as shown in Figure 11.

```
dynamic()
if (host makes a lot of long moves)
    Employ LU-PC.
else if ((frequently searched) and (low mobility))
    Employ LU-JU.
else if ((frequently searched) and (high mobility))
    Employ LU-PC.
else if ((Not frequently searched) and (high mobility))
    Employ LU-JU.
else Employ LU-PC.
```

Figure 11: *dynamic* - A dynamic location management algorithm

We present an example where a simple algorithm *dynamic* as shown in Figure 11 performs better than the *static* location management strategies. Simulations were performed for type (ii) environment. As stated earlier, a mobile host makes a lot of short moves in type (ii) environment. Thus, the dynamic location management algorithm *dynamic* makes a choice between $LU\text{-}JU$ and $LU\text{-}PC$ based on call frequency and mobility of the host. Figure 12 illustrates the mobility distribution of an user. The x-axis represents the time at which the user moves, and the y-axis represents the length of the move. Figure 13 illustrates the incoming call distribution for the user. The x-axis represents the time at which the call is made for the user, and y-axis represents the distance of the caller from the user. The value of $MTCI$ and $MTMI$ was chosen to be 10 units of time. For

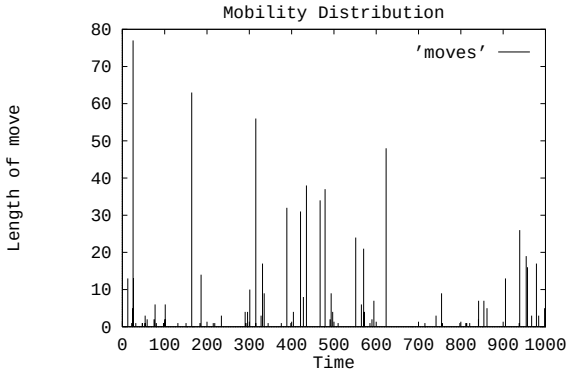


Figure 12: Mobility Distribution

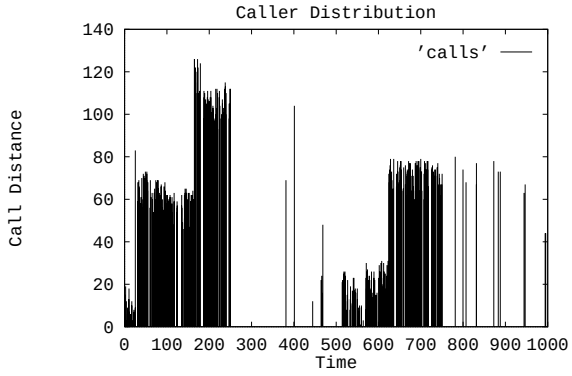


Figure 13: Call Distribution

this non-uniform call and mobility distribution, we evaluated the $LU\text{-}PC$, $LU\text{-}JU$ and *dynamic*

strategies. We evaluate the aggregate cost[9]. Figures 14, 15 and 16 illustrates the aggregate cost for *LU-JU*, *LU-PC* and *dynamic* strategies. For the given call and mobility distribution, results were obtained for different sizes of the move and call sets. It was observed that the minimum size of the move and call sets that was required for good performance of *dynamic* strategy was 7. Figure 17 illustrates the difference of aggregate cost between *dynamic* and *LU-JU* schemes. Figure 18 illustrates the difference of aggregate cost between *dynamic* and *LU-PC*. In Figures 17 and 18, negative difference implies that *dynamic* is better. As seen in Figure 17 and Figure 18, *LU-JU* performs poorly during periods of high-communication, and *LU-PC* performs poorly during periods of low-communication. However, on the average, *dynamic* performs better than both the schemes during periods of low and high communication, as illustrated in Table 1. Time interval 100.0-200.0 is the high communication period (107 calls or 1.07 calls per unit time). During this period, if the system designer uses *LU-JU* instead of *dynamic*, the network load (in terms of number of messages) will increase by 33%. Time interval 400.0-600.0 is the low communication period (91 calls or 0.45 calls per unit time). During this period, if the system designer uses *LU-PC* instead of *dynamic*, the network load (in terms of number of messages) will increase by 12%. Because, the input call distribution was equally distributed between periods of high and low communication over the *total* runtime, the advantage of using *dynamic* over *LU-PC* (4%) and *LU-JU* (17%) is not appreciable. However, the results show that a simple dynamic location management algorithm as shown in Figure 11 performs better than the *static* location management strategies for any call and mobility patterns.

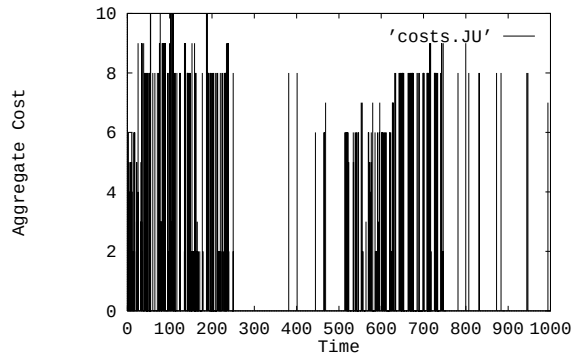


Figure 14: Aggregate Cost for *LU-JU*

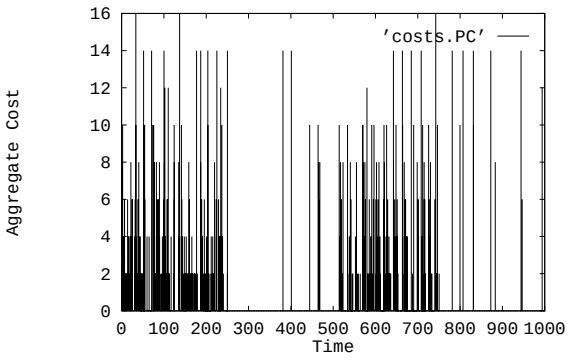


Figure 15: Aggregate Cost for *LU-PC*

[9] As stated earlier, we define aggregate cost as the sum of average update cost, average search cost, and the average search-update cost.

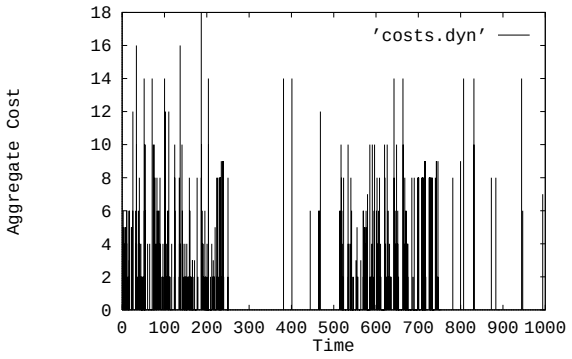


Figure 16: Aggregate Cost for *dynamic* - Size = 7

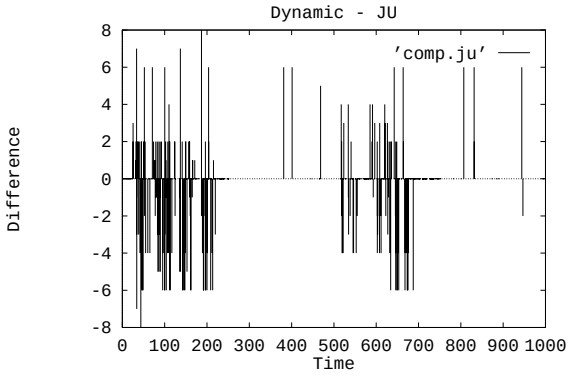


Figure 17: (*dynamic* – *LU-JU*)

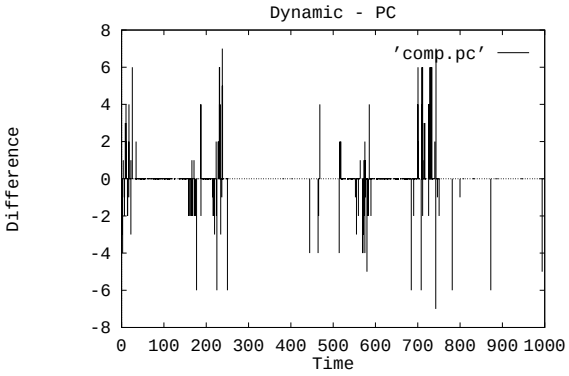


Figure 18: (*dynamic* – *LU-PC*)

Interval	# Calls	*LU-PC*	*LU-JU*	*dynamic*	Savings over *LU-PC*	Savings over *LU-JU*
100-200.0	107	3.32	4.08	3.1	6%	33%
400.0-600.0	91	3.36	3.02	3.0	12%	1%
0.0-1000.0	562	3.35	3.73	3.2	4%	17%

Table 1: Comparison of Average Costs for Non-Uniform distribution

7 Conclusions

This paper presents strategies for updates, search-updates, and a search protocol. A location management strategy is a combination of the search strategy, a update strategy, and a search-update strategy. Simulations were carried out to evaluate the performance of the various location management strategies. It was noticed that performing search-updates significantly reduced aggregate costs. For the logical network architecture assumed, it is seen that the *LU-PC* (combination lazy updates and path compression search-update) strategy performs better than the other strategies for most of the values of C and M. It is expected that *LU-PC* will perform well in other network models too.

Static location management uses one combination of search, update and search-update strategies throughout the execution. In order to obtain good performance using static location management, the system designer should a priori have a fair idea of the call and the mobility pattern of the users. The host behavior (call frequency, mobility) is not always available to the system designer. Thus, there is a need for a dynamic location management. In this paper we present preliminary ideas for dynamic location management. The basic philosophy behind dynamic management is that the past history of the system will reflect the behavior in the future and hence by keeping track of the past history and modifying the management strategy accordingly, one expects to perform well for any call and mobility pattern. Simulation results show that the performance of dynamic location management is better than static location management.

References

[1] T. Imielinski and B. R. Badrinath, "Mobile wireless computing: solutions and challenges in data management," Technical Report, Rutgers DCS-TR-296/WINLAB TR-49, Feb. 1993.

[2] 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.

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

[4] K. Keeton et.al., "Providing connection-oriented network services to mobile hosts," *Proc. of the USENIX Symposium on Mobile and Location-Independent Computing,* Cambridge, Massachussets, August 1993.

[5] Pravin Bhagwat and Charles. E. Perkins, "A Mobile Networking System based on Internet Protocol (IP)," *Proc. of the USENIX Symposium on Mobile and Location-Independent Computing,* Cambridge, Massachussets, August 1993.

[6] J. Ioannidis et. al., "IP-based Protocols for Mobile Internetworking," *Proc. of ACM SIGCOMM,* 1991.

[7] J. Ioannidis and G. Q. Maguire Jr., "The Design and Implementation of a Mobile Internetworking," *Proc. of Winter USENIX,* Jan. 1993.

[8] Charles Perkins, "Providing Continuous Network Access to Mobile Hosts Using TCP/IP," *Joint European Networking Conference,* May 1993.

[9] D. J. Goodman, "Trends in Cellular and Cordless Communications," *IEEE Communications Magazine,* June 1991.

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

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

[12] F. Teraoka, Y. Yokote and M. Tokoro, "A Network Architecture Providing Host Migration Transparency," *Proc. ACM SIGCOMM Symposium on Communication, Architectures and Protocols,* 1991.

[13] H. Wada et. al., "Mobile Computing Environment Based on Internet Packet Forwarding," *Proc. of Winter USENIX,* Jan. 1993.

[14] M. Spreitzer and M. Theimer, "Providing Location Information in a Ubiquitous Computing Environment," Tech. Rept., Xerox PARC, 1993.

[15] Jean-Paul Linnartz, "Narrowband Land-Mobile Radio Networks," Artech House, 1993.

[16] Wall Street Journal Reports, "Telecommunications", February 11, 1994.