Adaptive, Distributed Location Management in Mobile, Wireless Networks

Kevin Lee Sanjay Jha Nirupama Bulusu

{kevinl, sjha, nbulusu}@cse.unsw.edu.au

Abstract – **Location management refers to the problem of updating and searching the current locations of multiple mobile nodes in a wireless network. To make location management efficient, the sum of update and lookup costs of the location database must be minimized. We hypothesize that previously proposed location management techniques rely on fixed location databases, and are therefore unable to fully exploit the knowledge of user mobility patterns in the system so as to achieve this minimization.**

In this paper, we present a novel location management architecture that has the ability to react dynamically to user pattern changes by having *mobile* location databases. The key idea is to use a hierarchy of agents that act as location databases that can rapidly replicate to other base-stations in response to a change in user patterns. We provide an analytical cost model for the location management problem and validate our hypothesis through simulation studies. We measure the algorithm performance according to the algorithm's ability to handle location update and call finding, which are the primary objectives of our location management problem and demonstrate its effectiveness.

I. INTRODUCTION

Fundamental to the efficient operation of future Personal Communication Services (PCS) networks [1], self-organizing mesh networks [2,3,5] and emerging application domains such as pervasive location-aware computing and sensor networks is an efficient scheme for *location management*. Location management refers to updating and searching the "whereabouts" of mobile nodes inside a network. It involves registering and updating location information of mobile nodes (users) as well as searching their current location.

Various location management schemes have been designed to lower costs and improve the time efficiency of these searches and updates [11,12,16,17,19,20]. There are also many techniques that can be added to these schemes to further lower the search and update traffic. However, most existing systems rely on fixed sets of location databases for location management. These databases are inflexible and immobile; therefore, the system cannot readily take advantage of the user mobility patterns that are present in virtually all wireless networks. Furthermore, in many systems, the databases are not distributed in nature, causing reliability problems that are common to centralized systems.

Moreover, in recent years, there has been extensive research on novel self-organizing network architectures in the Mobile Ad Hoc Networks (MANET) community. The Terminode project [2] looks at developing a wide-area, autonomous, self-organized, wireless multimedia network that is totally independent of any fixed infrastructure. The Grid project at MIT [3] deployed a test-bed network composed of base-stations in cars (called CarNet) [4]. The Multihop cellular architecture [5] provides localized Ad-hoc networking within a cell, where mobile hosts within the cell help each other to forward packets to the base-station. By using multi-hopping, the cellular architecture can expand cell coverage while maintaining the transmission range of the base-station. For such infrastructure-less self-organizing networks, it is hard to maintain a fixed or centralized location database.

Location management schemes typically use two extremes. One extreme is up-to-date and exact location information at all sites, requiring huge bandwidth for each location update but minimizing lookup cost. The other extreme is storing no location information at any site. This does not require any update but the lookup cost will be prohibitive. The choice is to find a point between these two extreme approaches, where the costs of update and lookup are balanced [1]. In this paper, we propose an alternative design for location management with the following design goals:

- *Tunable*: The system can be readily tuned to reflect a particular user pattern. It exploits the user mobility patterns and the uneven regional distribution of mobiles to achieve efficiency.
- *Flexible*: The system can provide greater flexibility in terms of deployment of location information so as to achieve a balance of update and lookup costs.
- *Adaptive*: Location management algorithms that have the power to automatically configure the system to a desirable balance. It automates the tracking of user patterns, and can take advantage of a change in user patterns to the extent of the fluctuation during the hours on a day.

Our design depends on exploiting the user mobility patterns, that is, the unique intrinsic characteristics of the mobile user to optimize the cost of location management. A user mobility pattern can be described by two notions - *locality* and *regional call-to-mobility ratio*.

- *Locality of movement* refers to the user pattern where the mobile hosts usually stays in a fixed set of locations. This occurs mainly because of the daily habits of the mobile hosts users: most mobile host users spend the majority of their time at their home or workplace. *Locality of call distribution* characterizes the distribution of callees and callers. In real life, users exchange calls with a fixed group of people such as friends and family frequently. This, combined with the locality of movement, means that most calls from mobile hosts will be generated from and to a number of regions, such as the home and office of the user.
- *Regional Call-to-Mobility Ratio* measures the volume of calls against the frequency of location change. Low call-to-mobility users change location frequently and make relatively fewer calls, whereas high call-to-mobility users remain static most of the time, but call other mobile hosts relatively frequently.

. Our goal is to optimize the location management system by moving it between the two extremes mentioned above (no location information at any sites and up-to-date location information at all sites)[1], so as to balance the update and lookup costs. There are two important aspects to our design.

The first aspect is the *placement* of location databases. The key idea here is to exploit the *locality principle* of the mobile hosts. The intuition here is that it is preferable to place the database in the vicinity of the caller or callee, to minimize the amount of update or lookup network traffic. In many situations, this will mean that more than one database is required. These duplicated databases, called *replicates*, are placed in the necessary locations. Caching uses this principle, by placing replicate entries in the vicinity of the likely callers.

The second major aspect contributing to the optimization is the *number of replicates* used. To further balance the search and update cost of location information, the number of replicates used is itself based on a function of the call-to-mobility ratio.

The key contributions of the paper are the following:

- We conjecture that previously proposed location management schemes are unable to exploit the full advantage of user patterns in the system because of the inflexibility that arises from having fixed databases, and propose a novel location management architecture that relies on mobile location databases. It uses a hierarchy of agents to help provide a flexible and distributed system, wherein the agents rapidly replicate to other base-stations in response to a change in user patterns.(Section II)
- We propose an analytical cost model to formulate the location management as an optimization problem, show it is NP-complete, and suggest an approximation algorithm (service ability) to solve the optimization problem. (Section III)
- We perform simulations to demonstrate the effectiveness of our location management algorithm in terms of its ability to handle location updates and call finding. (Section IV)

II. LOCATION MANAGEMENT ALGORITHM

In this section, we describe the network model we assume, the underlying system architecture for our location management scheme, and the algorithms to register mobile users with location databases, replicate agents, and update location information.

A. Network model .

Our system is designed for a cellular architecture; this network infrastructure consists of many base-station nodes in the *backbone network* that are referred to as **base-stations**. The base-stations are equipped with wireless transceivers that can communicate to the mobile hosts.

The area that is covered by a base-station's signal is called a cell. Each mobile host can directly communicate with a cell's base-station. The mobile hosts communicate with units other than the base-station through the base-station. When a mobile host moves from one cell to another, it must communicate through the new base-station. In this case, the point of attachment, and hence the location of the mobile host is changed.

The base-station that is covering the mobile host is referred to as the foreign agent of the mobile host, and its address is called the Care-of Address (COA) of the mobile host. In our system, location information is stored in the form of COAs.

B. Architecture of the system

In this design the location information is stored inside some location databases. These location databases reside in the base-station nodes of the infrastructure, and are not hardware entities. We call these location databases agents. In our architecture, there are many such agents, and each one stores the location information of a number of mobile hosts. We say that an agent is bound to a base-station if the agent is available at this base-station.

An agent is a table with each entry storing the Care-of Address (COA) of another entity. Therefore, when a host wants to locate another mobile, it can read from the table and know which base-station, that is, which

COA, it should forward the call towards. For example, in Fig.1: Agent A5 has four entries. Each entry stores the location information of a callee: 15.168.0.1, 15.168.0.2, and so on. The notation above

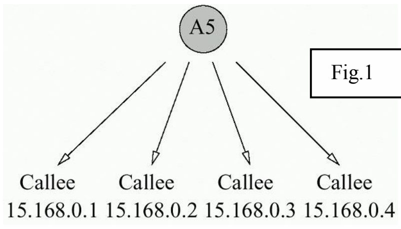


Fig.1

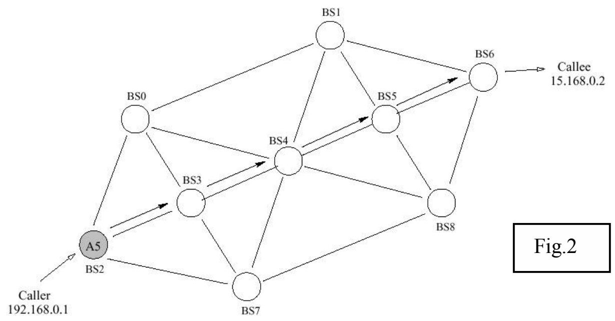


Fig.2

means that the agent A5 has a pointer (COA) for each callee. When a caller wants to call host 15.168.0.2 as in Fig.2, it must find out where it is. The base-station BS2, which is the foreign agent for the caller, looks up in A5, and finds out that the COA points directly to BS6, therefore it forwards the call to BS6 from BS2, and finally reaches the callee. In this case, the agent A5 is said to have a binding to the base-station BS2.

However, sometimes, it may happen that A5 is not bound to BS2, such as in Fig.3. The base-station BS2 would not know where to forward the call. There are two ways to find the callee. One way is to have location information about the callee, another way is to have location information about the agent A5 such that it can be reached. Suppose that we have another agent A1 that stores the location information of A5, and it is bound to the base-station BS2: In Fig.4, the agent A1 does not know where the callee is, but it knows where the agent for callee A5 is: it is bound to the base-station BS2. BS2 can then forward the call to BS8, where the location of the callee is known, and it is redirected towards the callee. Sometimes the location of A1 is not known. We then require another agent A0 to locate it, so it is quite likely that the call must be forwarded three times before it can reach the callee, as in Fig.5.

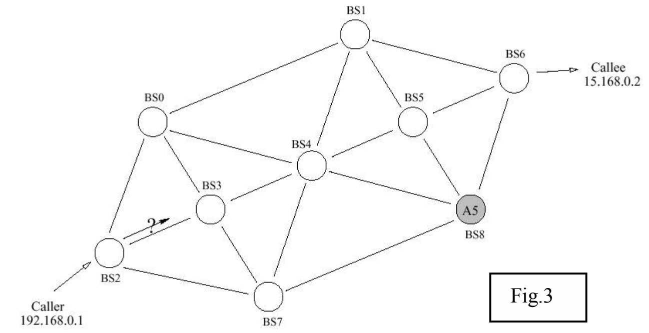


Fig.3

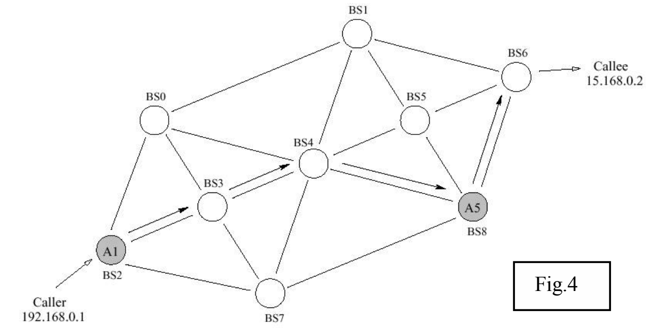


Fig.4

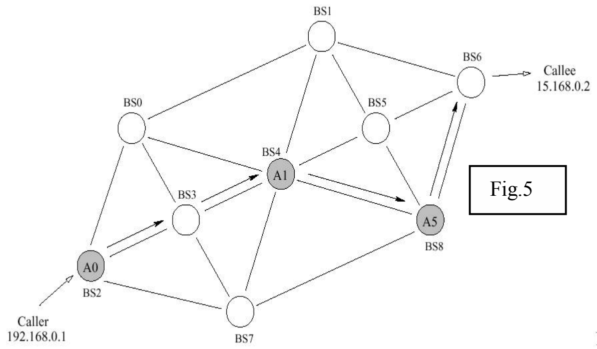


Fig.5

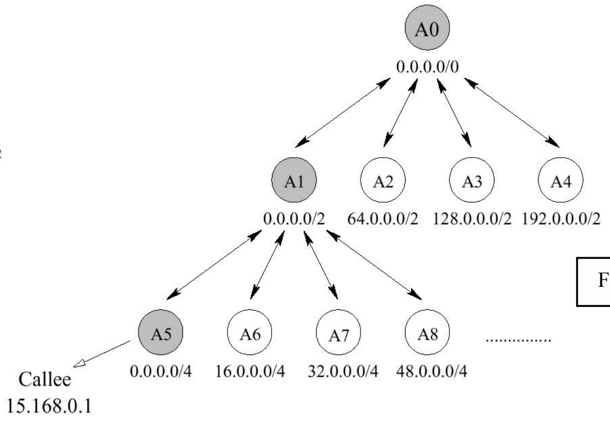


Fig.6

In our architecture, each agent stores more than one entry. An agent such as A0 can store the location information of many other agents, resulting a tree-like structure between different agents: Here, we say that A5 is a child of A1, and A1 is the parent of A5, similarly for A0 and A1. In our proposed architecture, the number of children ***b*** an agent can have is fixed. This gives a tree structure with branching factor ***b***. The leaf node of the tree stores the location information of the mobile hosts, the nodes on the upper level store the location of the location databases directly below them. Each agent in the tree has an individual unique netmask, so that mobile hosts that match a particular agent's netmask will be in the sub-tree of this particular agent. An agent can move freely within the network, by binding to the base-station it is migrating to. Therefore, it is possible that more than one agent has bound to a single base-station. Note that this design of our hierarchical tree is in fact a logical structure as opposed to the actual networking structure. This means that the tree exists logically within the actual backbone network that provides interfaces between physical base-station nodes.

C. New Mobile Registration

The agent hierarchy originally starts with only one agent, the root agent. It stores the location information of all mobiles, as in Fig.7, assuming there are only four mobiles in the whole system: When new mobile hosts are added into the system, the agent expands to form an agent hierarchy. For example, we want to add another mobile 192.168.0.2. The new mobile number is sent to the root of the agent hierarchy. The root forwards the registration to the child agent with the netmask that matches the mobile number. In this manner the number follows the path in the tree determined by the unique netmasks to the leaf agent of the hierarchy. If the leaf agent is not full, the leaf agent adds a location information entry for this mobile. If the leaf agent is already full, that is, the leaf agent is already hosting a number of mobile hosts equal to branching factor of the tree, the leaf agent will expand. In the expansion process, a number of new agents is produced equal to the branching factor of the tree, as in Fig.8.

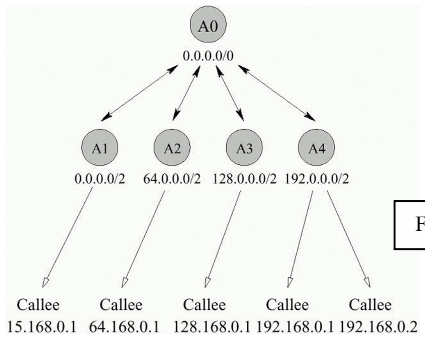


Fig.7

D. Agent Replication

From Fig.8, it can be observed that if callers are at similar physical locations, they often share close-to-common physical pathways to the same callee. What we mean by “similar physical locations” is that callers are residing in different base-stations that are close to each other, for example, BS2 and BS7. By observing the logical pathways of the two callers, we can see that both callers are in fact accessing two copies of the original root agent because of agent replicates. Thus in effect, even though the physical paths of the two callers to the same callee are different, the logical path that both callers took was in fact the same as in Fig.9.

The importance of this idea is the way in which the logical tree is “embedded” in the physical network allowing the two structures to relate to each other to seek optimal solution. Replication of agents can be applied to all agents in the hierarchy to shorten the physical paths in the network. By exploiting the advantage of this relationship between the two structures, we can group callers at similar physical locations and similar IP addresses so that callers in the group share exactly the same logical path, starting with virtually the same root agent and down the tree reaching the callee. By applying this grouping, we can significantly reduce the number of agents along the path.

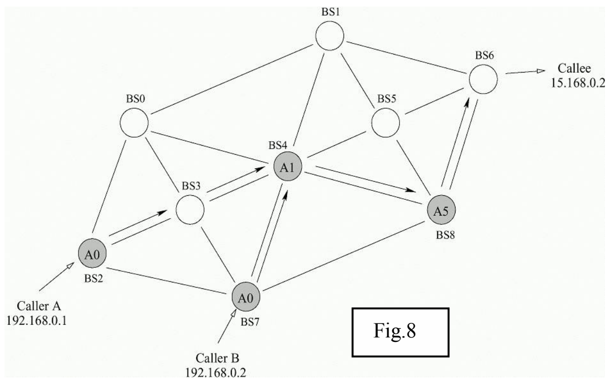


Fig.8

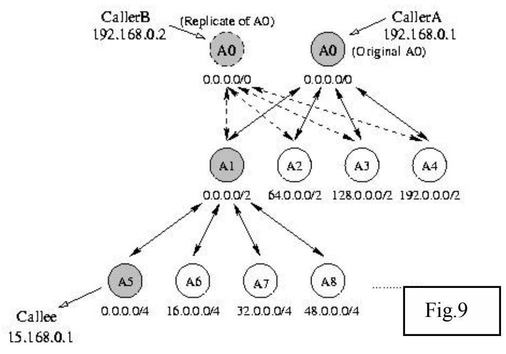


Fig.9

E. Updating Location Information

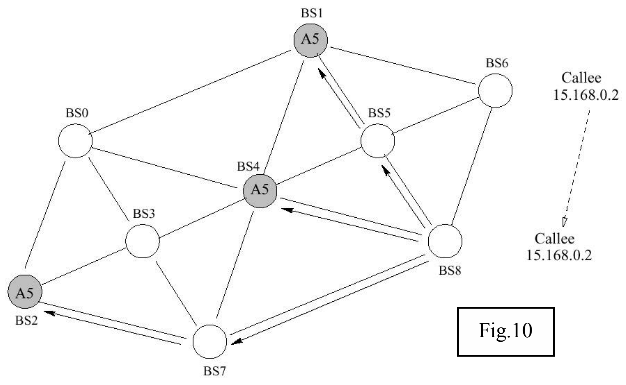


Fig.10

When a mobile moves, the location information must be updated. To improve efficiency, every mobile knows the locations of its parent agents, including the replicates. For example, in Fig.10. The parent agent of the mobile 15.168.0.2 is A5. When the mobile moves from BS6 to BS8, it will send notification to its parent agent. Since the mobile knows where all the replicates of A5 are, the update messages are sent directly to them. In this case, three messages are sent, to BS4, BS2 and BS1. With increasing number of replicates, the update cost of location management increases.

III. PERFORMANCE ANALYSIS

In this section, we analytically model the performance of our location management scheme. Two important factors contribute to the overall performance of our algorithm – *number of agent replicates* and *location of agent replicates*.

A. Cost Model

The cost model for the optimization problem is an adaptation of [6], originally developed in the context of Content Distribution Networks (CDNs). We consider the network where the nodes are the base-stations, and we denote each node by a number i, with $i \in \{1, 2, ..., I\}$, where I is the total number of base-stations in the network.

There are J replicates of the same agent, and a request probability p_{ij}, $j \in \{1, 2, …, J\}$, which is the probability that base-station i would choose to forward a location information request to agent j.

There is a request rate λ_i, defined as the number of requests that will originate from base-station i to one of the replicates. Which replicate is picked is determined by p_{ij}. There is an update rate μ_i defined as the number of updates that will originate from base-station i to the replicates in a given time period. The matrix of all p_{ij} is denoted by **p**. The update rate of a base-station to every replicate is the same, as all of the replicates must be updated when a mobile changes its point of attachment to the network.

The values of the parameters λ_i and μ_i are subject to the user patterns locality principle; λ_i is influenced by the locality of calls while μ_i is affected by the locality of movement. The result is an uneven distribution of λ_i and μ_i across the network. In this cost model, the cost is calculated in terms of the number of hops that each request or update must travel.

Now, let us assume that the base-station will always pick the replicate that is closest to itself when requesting location information. That is,

$$p_{ij} = \begin{cases} 1 & \text{if replicate j is the closest replicate to base-station,} \\ 0 & \text{otherwise.} \end{cases}$$

The matrix is denoted by **p**.

There is a distance between the base-station i and the closest replicate of agent j, measured by the number of hops in the shortest path between base-station i and the host base-station of that particular replicate. The average number of hops that a lookup must traverse from Base-Station i is d_{ij}, where j is the closest replicate to base-station i and d_{ij} is the shortest distance to the host base-station of replicate j from base-station i. This nearest replicate is either in base-station i or another base-station and the average number of hops that an update must traverse from base-station i is

$$\sum_{j=1}^{J} d_{ij}$$

where d_{ij} is the distance between base-station i and replicate j.

The cost, measured by the total hop counts incurred for a particular agent from a particular base-station is therefore

$$C_i = \sum_{j=1}^{J} p_{ij}\, \lambda_i\, d_{ij} + \sum_{j=1}^{J} \mu_i d_{ij} \qquad (1)$$

Here, it is assumed that each base-station has infinite storage capacity. It can hold as many agents as it wants.

Let $\Lambda_{ij}(\mathbf{p}) = p_{ij}\, \lambda_i$

The total location management cost for all base-stations and all replicates is then

$$\boldsymbol{C}(\mathbf{p}) = \sum_{i=1}^{I} \left(\sum_{j=1}^{J} \Lambda_{ij}(\mathbf{p})\, d_{ij} + \sum_{j=1}^{J} \mu_i d_{ij}\right) \qquad (2)$$

Notice that the ratio between λ_i and μ_i is influenced by the call-to-mobility ratio. Frequency of mobile movement determines the update rate μ_i while the frequency of calls determines the request rate λ_i. The update rate μ_{ij} is controlled by the number of replicates inside the network. As more replicates are installed inside the network, the request rate to each individual agent decreases. This change in request rate is denoted by

$\Delta\, p_{ij} = p_{ij1} - p_{ij0}$

and the matrix of all $\Delta\, p_{ij}$ is denoted by $\Delta\, \mathbf{p}$. Note that the normal value of $\Delta\, p_{ij}$ is -1 if the newly installed replicate replaced replicate j to become the closest replicate to base-station i, if j is a newly installed replicate, $\Delta\, p_{ij}$ is 1 or 0, otherwise it is 0.

Therefore, the resultant change in the location management cost from the installment of a new replicate j is

$$\Delta \mathbf{C(p)} = \sum_{i=1}^{I} \left\{ \left[\sum_{j=1}^{J} \Lambda_{ij}(\Delta \mathbf{p}) d_{ij} \right] + \mu_i \, d_{ij} \right\} \tag{3}$$

From the above result, we can see that the net effect of installing a new replicate is an increase in the update cost from all base-stations $\mu_i \, d_{ij}$ and a decrease in lookup costs associated with all other replicates $\sum_{j=1}^{J} \Lambda_{ij}(\Delta \mathbf{p}) d_{ij}$ ($\Delta \, p_{ij}$ is negative in 1 or more base-stations).

From this result, it is clear that the problem is to balance these two costs such that the total cost of location management becomes a minimum.

B. Efficiency analysis

To make things simple, we assume that all the ancestors of α have the same number of replicates n. We also assume that these ancestor agents are scattered randomly around the network. Therefore, the chance of having a replicate in a base-station is $\frac{n}{I}$, where I is the number of base-stations. We also assume that the average hop count from any base-station to any replicate of any non-leaf agent is D_n for the topology.

The average cost of locating a particular leaf agent for is therefore:

$$(1-\frac{J}{I})D_n + (1-\frac{J}{I})(1-\frac{n}{I})D_n + (1-\frac{J}{I})(1-\frac{n}{I})^2 D_n + \ldots + (1-\frac{J}{I})(1-\frac{n}{I})^{L-1} D_n \tag{4}$$

The first term is the cost of requesting the agent one level directly above the leaf agent. On average, there is a $(1-\frac{J}{I})$ chance that the leaf agent is not inside the base-station and requires a request to the agent one level above, recall that J is the number of replicates of the leaf agent. The second term is the cost of requesting the agent two levels directly above the leaf agent. There is a $(1-\frac{J}{I})(1-\frac{n}{I})$ chance that both the leaf agent and the agent above it are not present in the base-station.

Without the cost of locating the location database (leaf agent) the average total request cost incurred from a call is $\sum_{j=0}^{J} \Lambda_{ij}(\mathbf{p})\, d_{ij}$, as derived above. This is the cost for a GSM system with simple replicates.

The average total request cost incurred by a call in our system originating from base-station i is therefore

$$\sum_{j=0}^{J} \Lambda_{ij}(\mathbf{p})\, d_{ij} + (1-\frac{J}{I})D_n + \quad (1-\frac{J}{I})\,(1-\frac{n}{I})D_n + \ldots + (1-\frac{J}{I})\,(1-\frac{n}{I})^{L-1}D_n \qquad (5)$$

It can be seen that the additional cost involved in locating the location information for the callee is dependant on the number of replicates the ancestor agents of the leaf agent has, the topology of the network and the number of replicates the leaf node itself has. Since $J = \dfrac{b^L - 1}{b-1} \approx b^L$, a larger branching factor implies a smaller cost. The cost function of (5) is also inversely proportional to the branching factor.

C. Replication Heuristic (Service-Ability Algorithm)

The algorithm we study here attempts to provide an approximation to this optimization problem using a threshold value. We term it the *service-ability algorithm*. It uses the threshold to determine how many base-stations a replicate can serve. This algorithm is initiated by an agent.

Service-Ability Algorithm for a particular agent

```
topology
$i \leftarrow 0$
agent j
while topology is not empty do
    tmp
    tmpMetric ← -1
    for  all the base-stations b inside the topology do
        metric = 1 + CalculateMetric( b, threshold, 0, j)
        if  metric > tmpMetric do
```

```
                tmpMetric ← metric
                tmp ← b
            endif
            $i \leftarrow i + 1$
        endloop
        replicate j at tmp
        delete tmp from topology
    endloop

CalculateMetric (base-station, threshold, hop, j) {
    metric ← 0
    for all the neighbours n of base-station do
        if  $f$ (the call rate $p_{ij}$, hop + 1) ≤ threshold
        then        // $f$(x,y) is the service ability
                // function
                metric ← metric + 1
                metric ← metric + CalculateMetric (n, threshold, hop+1, j)
        endif
    endloop
    return metric
}
```

The advantage of this algorithm is that it does not place an arbitrary limit on the number of replicates that an agent can have. Furthermore, it ensures that all base-stations are adequately serviced, depending on the request rate p_{ij} and the distance from a replicate. The value of the threshold can be easily modified to control the number of replicates.

Complexity of this algorithm

Assume that the service ability function f runs at O(1) complexity. Also assume that the distance matrix is already computed for routing purpose for each node, and reading the d values runs at O(1) complexity. Given these assumptions, it can be deduced from the pseudo code that the complexity of the algorithm is $O(n^2)$.

IV. RESULTS AND ANALYSIS

The simulation shows that the system is efficient when there is a low call-to-mobility ratio, high locality of calls, high locality of movement and rapidly changing user patterns. An important parameter in the system is the branching factor of the agent hierarchy.

The simulated system consists of 1,000 mobiles, 10,000 connections (calls between mobiles), within a fixed network of 49 base-stations. There is also a repeativity R, which is the calling set size of each mobile. This is used to simulate the locality of calls. It is assumed that each user only calls a fixed set of mobiles. The size of this set is limited by R. The default value for R is 5.

A. Branching Factor

In Fig. 11 & 12, it is observed that as the branching factor increases, the average number of hops decreases. As the tree becomes flatter, the average number of hops that a request must travel from the root agent to the leaf agent decreases. For example, when the branching factor is 256, the agent hierarchy has only two levels. This results in an average hop count of 3.8352 as against 4.26 when the branching factor is 16. This corresponds to the result in the ***Performance Analysis*** section Equation (5).

B. Threshold

Fig. 13 & 14 show the threshold used in the algorithm against the average hop count. A larger threshold means that an agent is likely to service a larger number of base-stations resulting in smaller numbers of replicates in the system. When the branching factor is 16, the number of hop counts is around 7 for a threshold of 20. This reflects a network without any replicates. As the threshold is lowered, the average hop count decreases dramatically. When the threshold becomes one, every agent has a replicate in every base-station in the network, the hop count is lowered to 1.8 when the branching factor is 16. The decrease in hop count is steeper than the decrease in threshold. At the same time, the number of agents increases to 3578 when the threshold is one. The update cost of each mobile movement for the network is nearly 10 times (the average number of agents increased by nearly 10 times that of a network without any replicates. This shows that, for low call-to-mobility ratios, a high threshold would be appropriate, while for high call-to-mobility ratios, a low threshold would be more efficient.

By comparing the hop count and number of agents, it is observed that the replicates can improve the average hop count dramatically. The threshold for this algorithm can be chosen to suit a particular system.

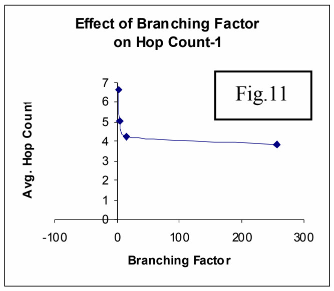


Fig.11

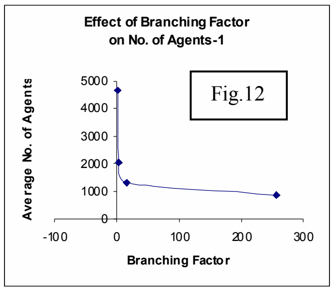


Fig.12

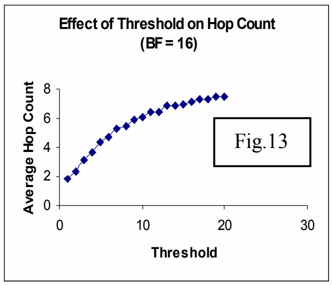


Fig.13

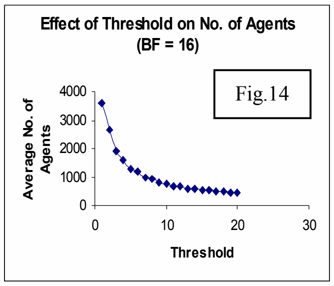
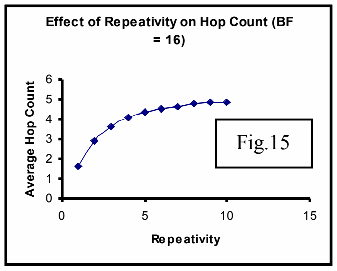
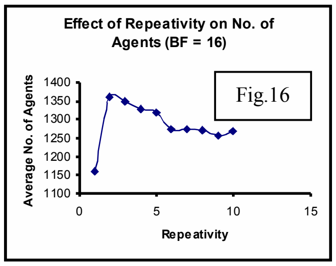


Fig.14





C. *Calling Set Size*

This is to model the locality of call, and captures the fact that most users call a few numbers (that of workmates, family) frequently. In Fig. 15 &16, it is shown that the system takes advantage of this locality. When the calling set is relatively large, the average hop count is much higher. The average hop counts drops by 66.4% to 1.64 when the branching factor is 16. The improvement in hop count was achieved with virtually no increase in the number of agents inside the system. This shows that the system actually migrates to the new base-stations to achieve the additional efficiency. When the repetitiveness is low, it is easier to find redundant agent replicates to remove from the system, this results in a low update cost since that the update cost of location information is proportional to the number of agents in the system.

V. CONCLUSION AND FUTURE WORK

Location management is fundamental to the efficient operation of cellular networks, mesh networks and pervasive networks. To make location management efficient, the underlying optimization problem is to find the point where the sum of the update costs and lookup costs of a location database are minimized. Previous work has proposed several general approaches to improve the cost of location management. Amongst them, placing replicates at suitable locations is the most important approach. Most of these approaches exploit the pre-existing user mobility patterns such as locality of movement, locality of calls and the call-to-mobility ratio to minimize costs. We conjecture that these location management techniques are unable to exploit the full advantage of user patterns in the system and achieve the optimization because of the inflexibility that arises from having fixed location databases.

To address this issue, we proposed a novel design that uses a hierarchy of agents to help provide a flexible and distributed location management system. These agents act as location databases that can rapidly replicate to other base-stations in response to a change in user patterns.

We proposed an analytical cost model for this location management problem. From this cost model, we proved that this optimization problem is NP-hard. The cost model further suggests that in the proposed architecture, the cost of locating a leaf agent can be made quite small. We also examined algorithms (service ability algorithm) to provide approximations to this optimization problem. Simulation results demonstrate that the proposed system can rapidly react to a change in user patterns.

This proposed architecture is far from complete. Among future problems is the development of a protocol to execute the service ability algorithm and a demonstration of the architecture on an unreliable network and evaluating its performance. The system can be improved if the selection of the threshold value can be automated. The values of threshold might be different for different levels of agents in the agent hierarchy. It is also possible to use multicasting to update location information.

Although key ideas are illustrated in the context of a cellular networks, the techniques can be applied broadly to any application domain involving mobile users/devices in wireless networks.

VI. REFERENCES

[1] E. Pitoura and G. Samaras. Locating Objects in Mobile Computing. *IEEE Transactions on Knowledge and Data Engineering*, 13(4): 571–592, 2001.

[2] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J-P. Hubaux, J-Y. Le Boudec. Self-Organization in Mobile Ad-Hoc Networks: the approach of Terminodes, *SSC Technical Report no. 2001/0xx on Communication Magazine, June 2001*

[3] CarNet Project. http://www.pdos.lcs.mit.edu/grid/

[4] Robert Morris, John Jannotti, Frans Kaashoek, Jinyang Li, and Douglas S. J. De Couto . CarNet: A Scalable Ad Hoc Wireless Network System: *In the Proceedings of the 9th ACM SIGOPS European workshop: Beyond the PC: New Challenges for the Operating System, Kolding, Denmark, September 2000.*

[5] Douglas S. J. De Couto, Daniel Aguayo, Benjamin A. Chambers, and Robert Morris. Performance of Multihop Wireless Networks: Shortest Path is Not Enough, *Proceedings of the First Workshop on Hot Topics in Networking (HotNets-I), Princeton, New Jersey, October 2002.*

[6] J. Kangasharju, J. Roberts, and K.W. Ross. Object replication strategies in content distribution networks. In Proceedings in WCW'01: Web Caching and Content Distribution Workshop, Boston, MA, *June 2001.*

[7] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving Objects Databases: Issues and Solutions. In Proceedings of the 10th International Conference on Scientific and Statistical Database Management, 1998.

[8] V. Anatharam, M. L. Honig, U. Madhow, and V. K. Kei. Optimization of a Database Hierarchy for Mobility Tracking in a Personal Communications Network. Performance Evaluation, 20:287-30, 1994.

[9] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In Proceedings of IEEE Infocom, Anchorage, AK, April 22-26, 2001.

[10] B. R. Badrinath, T. Imielinski, and A. Virmani. Locating Strategies for Personal Communications Networks. In Proceedings of the 1992 International conference on Networks for Personal Communications, 1992.

[11] G. Cho and L. F. Marshall. An Efficient Location and Routing Schema for Mobile Computing Environments. IEEE Journal on Selected Areas in Communications, 13(5), June 1995.

[12] Y. B. Lin. Determining the User Location for Personal Communications Service Networks. IEEE Transactions on Vehicular Technology, 43(3), August 1994.

[13] R. Jain. Reducing Traffic Impacts of PCS Using Hierarchical User Location Databases. In Proceedings of the IEEE International Conference on Communications, 1996.

[14] M.van Steen, F.J. Hauck, P. Homburg, and A.S. Tanenbaum. Locating Objects in Wide-Area Systems. IEEE Communications Magazine, pages2-7, January 1998.

[15] N. Shivakumar, J. Jannink, and J. Widom. Per-User Profile Replication in Mobile Environments: Algorithms, Analysis, and Simulation Results. ACM/Baltzer Journal of Mobile Networks and Applications, 2(2):129-140, 1997.

[16] N. Shivakumar and J. Widom. User Profile Replication for Faster Location Lookup in Mobile Environments. In Proceedings of the 1st ACM International Conference on Mobile Computing an Networking (Mobicom ’95), 161-169, October 1995.

[17] J. Jannink, d. Lam, N. Shivakumar, J. Widom, and D.C. Cox. Efficient and Flexible Location Management Techniques for Wireless Communication Systems. ACM/Baltzer Journal of Mobile Networks and Applications, 3(5):361-374, 1997.

[18] O. Wolfson, S. Jajodia, and Y. Huang. An Adaptive Data Replication Algorithm. ACM Transactions on Database Systems, 22(2):255-314, June 1997.

[19] S. Rajagopalan and B. R. Badrinath. An Adaptive Location Management Strategy for Mobile IP. In Proceedings of the 1st ACM International Conference on mobile Computing and Networking (Mobicom ’95), Berkeley, CA, October 1995.

[20] J. S. M. Ho and I. F. Akyildiz. Local Anchor Scheme for Reducing Signalling Cost in Personal Communication Networks. IEEE/ACM Transactions on Networking, 4(5), 1996.

[21] P. B. Mirchandani. The p-median Problem and Generalizations, in Discrete Location Theory, P. B. Mirchandani and R. L. Francis eds., Wiley-Interscience, New York, 1990, pp. 55- 117.