Adaptive Location Management in Mobile Networks *

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

October 13, 1994

Abstract

Static location management uses one combination of *search*, and *update* strategy throughout the execution [1]. 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 of *search* and *update* strategy which performs best for the given values of communication and mobility. However, the host behavior (communication frequency, mobility) is not always available to the system designer. Thus, there is a need for *adaptive* location management. In this paper we present a scheme for *adaptive* location management. The basic assumption behind *adaptive* 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 *adaptive* location management is better than static location management.

Technical Subject Area : PCS, PCN, and Mobile Systems & Networks.

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

1 Introduction

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

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. 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.

Numerous location management strategies have been proposed in the recent years [1, 3, 4]. These location management strategies are mainly a combination of a *search*, a location *update* strategy, and a *search-update* strategy. The results show that there is not one combination that outperforms others for all values of call frequency (C) and mobility (M) values. As shown in Figure 1a, we expect zones in the M-C plane, where one scheme will outperform others for the call frequency and mobility values in the zone. 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. However, the host behavior (communication frequency, mobility) is not always available to the system designer. Thus, we feel that the location management strategies with the greatest potential benefit are *adaptive* in the sense they react to changes in the host behavior (call frequency, mobility).

Adaptive strategies can range from simple to complex in their acquisition and use of host behavior information. The potential advantage of a complex policy is the possibility that the system will be operating at the *knee point* (lowest search cost and update cost possible) of the update and search tradeoff curve. The potential disadvantage is the overhead cost.

The goal of this paper is not to propose a specific adaptive location management strategy, but rather to address the ineffectiveness of the static location management strategies. We show that extremely simple adaptive location management strategies, which collect very simple amounts of data (host behavior) and which use this information in very simple ways, yield significant performance improvements over the static location management strategies.

2 Overview of Static Location Management Strategies

In this paper, we will try to develop a adaptive location management scheme based on our earlier work [1]. We had proposed a *search* strategy and various strategies for location *updates* and *search-updates*. We will give a brief overview of the strategies in the following. Details can be found in [1].

2.1 Search Strategy

If a host h in cell src wants to call another host h', h has to know the location of h'. This requires that host h search for host h'. The search strategy in the absence of an explicit *home* location server is as follows. If the mobile support station (MSS) of src 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 the 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 goes on till there is a location server which has location information for h'. 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.

2.2 Update Strategy

The strategies for updating the location information at the location servers and the mobile support stations (MSS), due to the movement of the host are as follows.

• Lazy Updates (LU) : This is the simplest update scheme. Updates take place only at the source and destination mobile support stations. A forwarding pointer is kept at the source mobile support station.

• Full Updates (FU) : Upon a move, apart from the mobile support stations involved, location updates take place in all the location servers located on the path from the mobile support station to the root, both at the source and the destination cells.

• Limited Updates (LMU) : This strategy is a compromise between the two previous strategies. Update in the location information takes place at a limited number of level of location servers in the tree. Here updates occur at $l < H$ number of levels of location servers on the path to the root. Updates at these location servers are similar to the FU scheme. The location servers on the path, but, at levels higher that l are not updated.

2.3 Search-Update Strategy

Location management becomes more efficient if the location updates take place also 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' at h after a successful search, so that in the future if h calls again, the search cost is most likely to reduce. Following are the strategies to update location information upon a search.

• Lazy Update (LU) : In this strategy, there are no location updates. But, the forwarding pointers corresponding to the destination host, is updated at the mobile support stations on the search path.

• Jump Update (JU) : In this strategy, a location update takes place only at the caller's mobile support station.

• Path Compression Update (PCU) : In this strategy, upon a successful search, a location update takes place at all the location servers and mobile support stations on the search path.

2.4 Results

The performance parameter of interest is the aggregate cost per operation, which is the sum of average update cost upon a move, average search cost, and the average update cost upon a search.

Simulations were performed for two types of environments : (i) arbitrary moves and arbitrary callers, (ii) short moves and a set of callers. Type (ii) is the closest to real life mobile environments. Users are expected to make a lot of short moves, and are expected to receive from a specific set of callers (for e.g. family, business colleagues). For the sake of brevity, we will denote :

• LU-LU as the combination of lazy update upon move, and lazy update upon search.

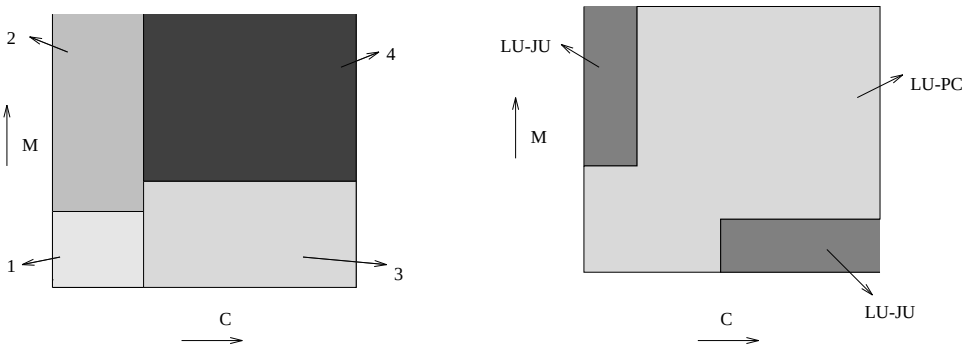
• LU-JU as the combination of lazy update upon move, and jump update upon search.

• LU-PC as the combination of lazy update upon move, and path compression update upon search.

Likewise FU-LU, FU-JU, FU-PC are the combination of full update upon move, and lazy update upon search, jump update upon search, path compression update upon search respectively. And, LM-LU, LM-JU, LM-PC are the combination of limited update upon move, and lazy update upon search, jump update upon search, path compression update upon search respectively.

The results show that there is not one combination that outperforms others for all values of call frequency (C) and mobility (M) values. This was evident in the Type (ii) environment. As shown in Figure 1b, 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 ideas and results for adaptive location management.



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

Figure 1: Partitioning of the M-C plane

3 Adaptive 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.

3.1 Theory

The problem in hand is to predict the present value of a given process in terms of its past values [7]. There are two relevant parts to this problem : **infinite past**, **finite past**. Infinite past deals with the estimation of a process $s[n]$ in terms of its entire past $s[n-k]$, $k \geq 1$. Infinite past will involve storing of all the past values. This will cause significant storage overhead. Therefore, infinite past is not a viable option. We will deal with prediction based on finite past. Finite past deals with the estimation of a process $s[n]$ in terms of its N most recent past values. Figure 2 is an example of such a predictor [7].

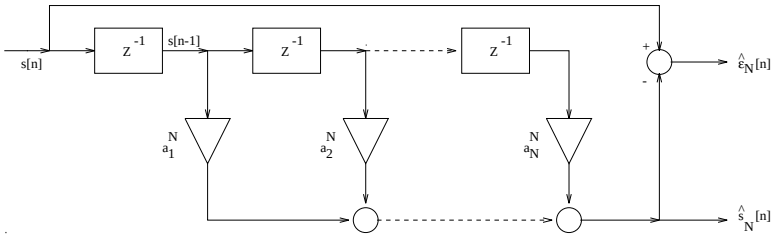


Figure 2: Finite Past Based Predictor

$$\hat{s}_N[n] = \hat{E}\{s[n]|s[n-k], 1 \leq k \leq N\} = \sum_{k=1}^{N} a_k^N s[n-k] \quad (1)$$

The estimate $\hat{s}_N[n]$ is called the *forward* predictor of order N. The superscript in a_k^N identifies the order. Generally, the object of a predictor is to minimize the mean square value of the forward prediction error $\hat{\epsilon}_N[n] = s[n] - \hat{s}[n]$. However, we simplify the design of the predictor used in this paper. The basic aim of this paper is to highlight the effectiveness of adaptive location management strategies, rather than determine which predictor will perform the best.

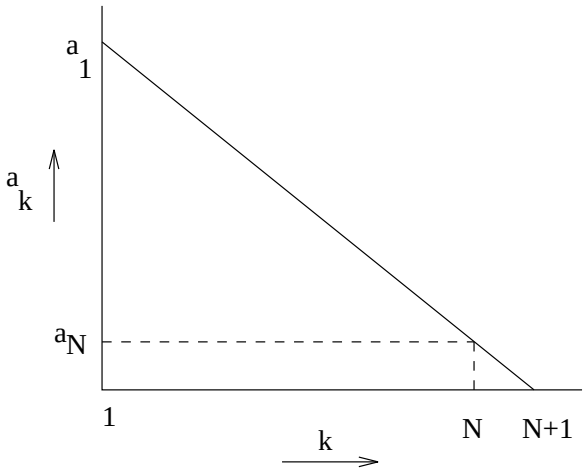


Figure 3: a_k^N as a function of k

Thus, we simplify equation (1) by setting $a_k^N = \frac{2(N+1-k)}{N(N+1)}$, $1 \leq k \leq N$, such that $\sum_{k=1}^{N} a_k^N = 1$. The distribution of a_k^N is as shown in Figure 3. Therefore,

$$\hat{s}_N[n] = \frac{1}{N(N+1)} \sum_{k=1}^{N} 2(N+1-k)s[n-k] \quad (2)$$

3.2 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* [1].

$C_u(h)$ 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 later.

3.3 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 [1].

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.

3.4 Mobility and Call Frequency

3.4.1 Determining Mobility

Let at time $t = \tau$, $M(h) = \{m_1, m_2, ..., m_N\}$, where $m_i = (src, t_N, dest)$, and $t_N \leq \tau$. Thus, m_i describes the move of host h that took place at time t_i 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 predicted time interval before the next move is $\triangle t_{(pred)} = \frac{\sum_{i=1}^{N} i \triangle t_i}{N(N+1)}$.

We assume a system parameter *maximum threshold move interval* ($MTMI$). If there are no moves (cell crossings) 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 low 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 $\triangle t_{(pred)} < MTMI$, the host has a high mobility, else if $\triangle t_{(pred)} \geq MTMI$, the host has a low mobility.

3.4.2 Determining Call Frequency

Let at time $t = \tau$, $S(h) = \{s_1, s_2, ..., s_N\}$, where $s_i = (t_{si}, h')$, and $t_{sN} \leq \tau$. Thus, s_i describes the call for host h from h' that took place at time t_{si}. We predict the time interval before the next call from **each** caller to host h. The predicted time interval before the next call from caller h', $\triangle t_{s(pred)}[h'] = \frac{\sum_{i=1}^{N'} i \triangle t_{si}}{N'(N'+1)}$, 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 an entry for h' in $S(h)$ set, the caller h' is assumed to be a non-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 $\triangle t_{s_{(pred)}}[h'] < MTCI$, the caller is a frequent caller, else if $\triangle t_{s_{(pred)}}[h'] \geq MTCI$, the caller is a non-frequent caller.

3.4.3 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.

3.5 An Example

In this section we will present an example algorithm for adaptive location management. It is for the network model assumed for static location management strategies [1]. The knowledge of Figure 1b, and the fact that LU-PC is the best scheme for long moves [1], 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 4.

```
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 4: *adaptive* - An adaptive location management algorithm

We present an example where a simple algorithm *adaptive* as shown in Figure 4 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 adaptive location management algorithm *adaptive* makes a choice between LU-JU and

LU-PC based on call frequency and mobility of the host. Figure 5 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 6 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

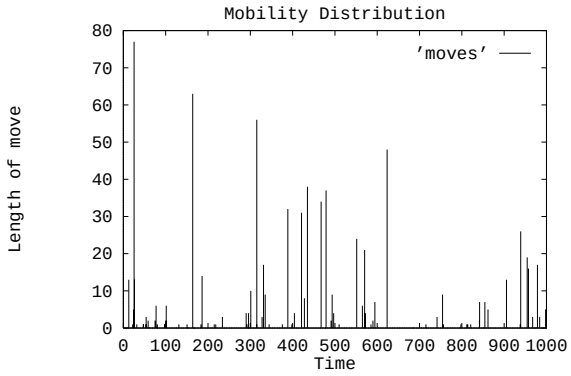


Figure 5: Mobility Distribution

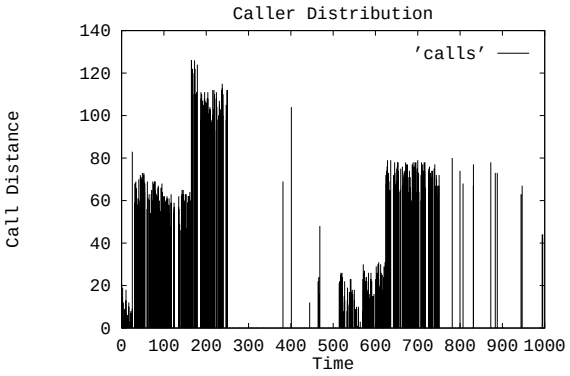


Figure 6: Call Distribution

this non-uniform call and mobility distribution, we evaluated the *LU-PC*, *LU-JU* and *adaptive* strategies. We evaluate the aggregate cost[1]. 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. *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 29%. 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 6%. 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* (5%) and *LU-JU* (23%) is not appreciable. However, the results show that a simple dynamic location management algorithm as shown in Figure 4 performs better than the *static* location management strategies for any call and mobility patterns.

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

Interval	# Calls	*LU-PC*	*LU-JU*	*adaptive*	Savings over *LU-PC*	Savings over *LU-JU*
100-200.0	107	4.77	6.04	4.7	1%	29%
400.0-600.0	91	4.26	4.1	4.02	6%	2%
0.0-1000.0	562	4.5	5.3	4.3	5%	23%

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

4 Conclusions

A location management strategy is a combination of the search strategy, a update strategy, and a search-update strategy. 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 adaptive location management. In this paper we present preliminary ideas for adaptive location management. The basic assumption behind adaptive 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 adaptive location management is better than static location management.

There are several issues deserving further study with respect to the deployment of adaptive location management strategies, such as effect of sophisticated prediction strategies, and the effect of alternative network architectures.

Acknowledgments

The authors wish to thank Prof. Dhiraj Pradhan for his helpful comments.

References

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

[2] P. Krishna, N. H. Vaidya and D. K. Pradhan, "Static and Dynamic Location Mana gement," Tech. Report 94-030, Dept. of Computer Science, Texas A&M University. (submitted to *Journal of Computer Communications*).

[3] 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.

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

[5] Amotz Bar-Noy et. al., "Mobile Users: To Update or not to Update ?," *Proc. of INFOCOM,* pp. 570-576, 1994.

[6] U. Madhow et. al., "Optimization of Wireless Resources for Personal Communications Mobility Tracking," *Proc. of INFOCOM,* pp. 577-584, 1994.

[7] A. Papoulis, "Probability, Random Variables, and Stochastic Processes," Third Edition, McGraw-Hill, Inc.