An adaptive location management strategy using forwarding pointers

Amal El Nahas and Noha Adly
Computer Science Dep. Faculty of Eng., Alexandria University 21544 Alexandria, Egypt

We present an adaptive location management strategy using forwarding pointers to locate mobile users in Personal Communications Systems (PCS). The proposed strategy aims at reducing the overall location cost by adapting itself according to the user mobility and calling patterns. The work is motivated by the observation that the set of callers initiating most of the calls for a mobile host is actually small and relatively stationary. Thus, a saving in the lookup delay can be realized by informing this set of callers at each move. This saving is significant for high Call to Mobility Ratios (CMR) but imposes excessive update messages for low CMR. A reduction in the number of updates can be made possible by using forwarding pointers and only informing the set of frequent callers whenever the pointers chain length reaches its maximum. This modification reduces the cost of updates for low CMR but at the expense of a slight increase in the lookup cost. The algorithm determines the frequent caller set dynamically using a simple on-line algorithm. Moreover, to tradeoff between lookup and update cost, it dynamically opts for the use of forwarding pointers according to the rate of mobility. We quantify the costs and benefits of the proposed strategy for a range of CMR values. It has been shown that, under certain assumptions, for low CMR, our strategy can result in 30% to 60% savings in the total cost, and for high CMR, the improvement ranges from 20% to 50%.

في هذا البحث نتقدم بنظام مطوع لإدارة المواقع باستخدام المؤشرات الأمامية في أنظمة الاتصالات الشخصية. النظام المقترح يهدف إلى تقليل التكلفة الكلية بالتكيف تبعا لحركة المستخدمين ونماذج الاتصال. النظام أسس على الملاحظة أن مجموعة المتصلين بمستخدم متحرك بصفة متكررة عادة تكون صغيرة وثابتة نسبيا. ولذلك فانه يمكن الإقلال من زمن البحث بإخطار هذه المجموعة عند كل حركة خاصة عند ارتفاع نسبة الاتصال إلى الحركة. ولكن هذا يؤدى إلى ارتفاع تكلفة التحديث. ولذلك فان النظام يتبع استخدام المؤشرات الأمامية ويخطر مجموعة المتصلين المتكررين عند وصول سلسلة المؤشرات إلى طول معين. هذا التعديل يؤدى إلى الإقلال من تكلفة التحديث خاصة عند انخفاض نسبة الاتصال إلى الحركة. النظام المقترح يحدد مجموعة المتصلين المتكررين ديناميكيا ولكي يوازن بين تكلفة التحديث والبحث فانه يتجه إلى استخدام المؤشرات الأمامية تبعا لمعدل الحركة. ولقد تم تقييم التكلفة والتحسين الناتج من النظام المقترح لمدى من نسبة الاتصال إلى الحركة وتبين أنه يؤدي إلى تخفيض في التكلفة الكلية بنسبة ٣٠% إلى ٦٠% عند انخفاض نسبة الاتصال إلى الحركة وبنسبة ٢٠% إلى ٥٠% عند ارتفاعها.

Keywords: Personal communication systems, Mobile computing, Location management, Forwarding pointers.

1. Introduction

Managing the location of mobile users as they relocate from one geographically bounded area, *zone,* to another is one of the challenging tasks in Personal Communication Systems (PCS). Location management strategies involve two basic operations: *update* and *lookup* operations. The *update* operation is required whenever the mobile host changes its location, and the *lookup* operation is required whenever the network attempts to deliver a call to the mobile host. Several techniques have been proposed to optimize the lookup and update operations, see [1,2] for a survey of those techniques.

The current Location management strategies implemented in the GSM and IS_41 standards are based on the *two-tier* architecture [3,4] where each user has a home database, named Home Location Register (HLR), that maintains his current location and is updated at each move. A Visitor Location Register (VLR) is maintained at each zone and stores a copy of user's location not at home and currently located at that zone. When a

user moves to a new zone, it updates its HLR, registers at the new VLR and deregisters at the previous VLR. When a call is placed from a remote caller to the mobile host (MH), the VLR at that caller is queried first. If the MH is not found, then the message is forwarded to the HLR, which sends it to the current location of the MH. This technique is known as the basic scheme.

In [5], it has been observed that while the potential set of callers for the MH maybe large, the set of callers that a given MH communicates most frequently with is very small, and relatively stationary. Further, it could be less costly to route calls addressed to the MH directly from the caller rather than through the HLR. Based on these observations, in [5,6] an adaptive scheme has been proposed where the mobile user informs all its frequent callers of its current address on every move. This technique is known as the informed scheme and it results in significant reduction in the lookup cost at the expense of increasing the update cost. They developed the concept of a working set of hosts for the MH and used an on-line algorithm that enables MH to dynamically determine its working set and tradeoff the lookup and update cost. For each caller, the MH decides whether it is more beneficial to use the informed scheme or the basic scheme. It has been shown that the scheme performs efficiently for high Call-to-Mobility Ratio (CMR), where CMR is defined as the number of calls addressed to the MH in a given time period divided by the number of moves made by the MH at that given time period.

A similar approach based on replication was presented in [7] for two-tier environment. They suggested to replicate user's location at certain sites to reduce the lookup latency and the communication cost if the savings due to local lookups exceed the update cost for maintaining the replicas consistent at every move. They developed an off-line minimum-cost max-flow based algorithm to determine the set of sites at which a user profile should be replicated given known calling and user mobility patterns. In [8], they extended the technique to hierarchical architecture by replicating user's profiles at internal nodes using an off-line algorithm, HIPER. The work in [9] extended HIPER by using on-line mobility and calling patterns estimates to select replicas placed at databases while satisfying certain constraints and dynamically migrate replicas to appropriate databases.

Although these algorithms showed good performance for high CMR, they performed poorly for low CMR due to the excessive update overhead. Since the number of updates is expected to increase dramatically in third generation PCS, location management schemes must consider reducing the overhead of update signaling.

In this paper, we present an adaptive scheme that adapts itself according to the user's mobility and calling patterns to reduce the total cost. Our strategy adopts the informed scheme using a simple on-line algorithm to determine the set of frequent callers that will be informed about user location at each move. Further, in order to reduce the cost of updates, the scheme deploys forwarding pointers for low CMR.

Forwarding pointer technique has been used previously to reduce the location update cost. In this technique, when the MH moves from one zone to another, no update message is sent to its HLR. Instead, a pointer is set up at the previous VLR pointing to the new one. When a call is placed to that host, the HLR is queried to determine the first VLR at which the user was registered, then the chain of pointers is followed until the MH is located. The lookup delay is kept low by bounding the maximum chain length to a predefined value.

A per-user forwarding strategy, deployed along with the basic scheme, is proposed in [10] for users with low CMR. This strategy shows an improvement in the update cost but at the expense of slightly increasing the call set-up time. In [11], forwarding pointers in two-tier architecture was adopted and extended by considering multiple HLRs. When the MH moves, the distributed HLRs are not updated, but a forwarding pointer is set as before. When a call is placed to the MH, it is directed to the nearest HLR that will then follow the chain of forwarding pointers until the MH is reached. Further, this particular HLR updates its pointer. This scheme attempts to reduce the load on HLR and

reduce the lookup cost without increasing the update cost.

Forwarding pointers have been proposed for the hierarchical architecture in [12,13] to reduce the cost of updates. Instead of updating all databases involved in the hierarchy, a forwarding pointer to the new location is set at the lower level database. This reduces the cost of updates since fewer internal databases are updated but the lookup cost increases since a number of forwarding pointers must be traversed before locating the callee. Several purging strategies were introduced to limit the lookup delay.

In our proposed scheme, for users with low CMR, forwarding pointer technique is used to avoid update messages sent to the user's HLR and the set of frequent callers at each move. This is shown to be efficient when the number of moves is relatively higher than the number of calls addressed to the MH.

We present a preliminary analysis of our proposed strategy. The analysis shows that, under certain assumptions, our strategy can result in 30% to 60% improvement in the total cost for low CMR and 20% to 50% for high CMR, compared to the basic scheme.

This paper is organized as follows. Section 2 presents our proposed strategy. A preliminary simplified analytical model evaluating the algorithm is presented in section 3. Section 4 presents some of the conducted experiments and discusses the obtained results. Finally, section 5 gives conclusions and ideas for future work.

2. The location scheme

The proposed location strategy is based on the idea of reducing the lookup cost by informing the frequent caller set directly about the moves of the mobile host, especially for high CMR. However, this will be on the expense of the increase in the move cost since the frequent callers are updated at every move. Therefore, the proposed scheme uses forwarding pointers when the mobility rate is high and updates the frequent callers only when purging the chain of pointers. It informs the frequent callers on every move only when the user's moves are relatively low.

We define the Frequent Callers Set (FCS) of a mobile user j to be the set of callers that j communicates most frequently with. Call and mobility statistics are collected for each user, and then, based on estimated user Local Call to Mobility Ratio (LCMR), callers are selected to join the FCS. The LCMR is the rate at which the user receives calls from a particular caller to the rate at which it moves between zones [10]. More precisely, $LCMR_{i,j}=C_{i,j}/M_j$, where $C_{i,j}$, is the number of calls made by caller i to mobile j in a given time period, and M_j, is the number of moves made by j in the same period of time. In contrast to off-line schemes, our strategy computes on line estimates of $LCMR_{i,j}$ and augments the FCS with callers that have sufficiently high LCMR. Consequently, the scheme can adapt to changes in user calling and mobility patterns by dynamically adjusting the FCS such that only frequent callers are always informed of the user's moves while other callers are forced to search. The adaptive scheme can be described in terms of its two main operations: **Move** and **Lookup** as follows.

2.1. Move operation

Consider a mobile j moving from VLR_x to VLR_y. Define the vector V_j maintained for user j and containing $C_{i,j}$ for all i that sent calls to j. The actions taken by VLR_y are as follow:

```
begin
  Compute CMR= Σ ∀i Ci,j/Mj ;
  if CMR ≥ α OR (CMR < α AND
  chain_length = K) then
     // Reevaluate the Frequent Callers Set
for all callers
   Old_FCS = FCS, FCS=Φ;
   ∀ i that has an entry in Vj do
      LCMRi,j = Ci,j/Mj;
      if LCMRi,j > β then
            Add i to FCS;
            if i ∈ Old_FCS then remove i
   from Old_FCS fi;
            fi;
   od;
      // Invalidate the information in sites
   that belonged to FCS but are not eligible
   any more
```

```
    if Old_FCS ≠ Φ then invalidate s ∀ s ∈
   Old_FCS fi;
  fi;
  if CMR ≥ α then  // low mobility
    update HLRj;
    purge forwarding pointers, if any;
    if FCS ≠Φ then update s ∀ s ∈ FCS fi;
  else    // high mobility, use forwarding
pointers
    increase chain_length by 1;
    if chain_length ≤ K then
        detect cycles and compress pointers
        set forwarding pointer at VLRx pointing
    at VLRy;
    else
        update HLRj setting it to VLRy;
        if FCS ≠Φ then update s ∀ s ∈ FCS
        setting them to VLRy fi;
        invalidate pointers in the purged
        chain;
        chain_length = 0;
    fi;
  fi;
end;
```

The actions taken by VLR_y depend on the call to mobility ratio (CMR). For low mobility ($CMR \geq \alpha$), forwarding pointers are not used and the HLR as well as the set of frequent callers are updated directly at each move. The main benefit here is to reduce the lookup cost, since the number of lookups exceed the number of updates. In case of high mobility ($CMR < \alpha$), and in order to reduce the cost of moves, forwarding pointers are deployed and neither the frequent callers nor the HLR are updated. In order to limit the increase in the lookup cost, the chain of pointers is allowed to increase up to length K. Beyond K, the chain is purged by updating the HLR and the set of frequent callers with the new location and a new chain of forwarding pointers is restarted. All existing pointers are invalidated and VLR_y becomes the first node in the new chain. Updating the FCS in this case is beneficial because subsequent lookups initiated from members of FCS can follow the chain of pointers without having to go through the HLR.

Before setting the forwarding pointers, cycles must be detected in order to avoid infinite loops during calls. Cycles can be easily detected, as suggested in [12,13] by checking whether a pointer for user j already exists at VLR_y. If so, then there is a path from VLR_y to VLR_x that needs to be purged. For example, consider a chain $VLR_a \rightarrow VLR_y \rightarrow VLR_b \rightarrow VLR_x$ and a move is made to VLR_y. Obviously, adding a pointer to VLR_y results in the chain $VLR_a \rightarrow VLR_y \rightarrow VLR_b \rightarrow VLR_x \rightarrow VLR_y$ and future calls will hang. Upon the move from VLR_x to VLR_y, a check is made whether an entry for j already exists in VLR_y. Since an entry is found, then there is a path from VLR_x to VLR_y that can then be purged. Thus, the chain of the above example becomes $VLR_a \rightarrow VLR_y$. For details on cycles detection and removal, the reader is referred to [12,13,14].

2.2. Lookup operation

The Lookup operation proceeds as follows. Consider a caller i placing a call to mobile user j. The caller i performs the following.

```
begin
   if j's location is validated at i then
       Search at that location;
    if not found then follow forwarding
    pointer fi;
     else
     Query HLRj;
        if not found then follow forwarding
   pointer fi;
          fi;
end;
```

2.3. Maintaining FCS

In order to determine whether a caller i can be classified as a frequent caller of j, $LCMR_{i,j}$ is considered, if it exceeds a certain threshold β, then i is a candidate to join FCS. Obviously, if β is set too high it takes large $LCMR_{i,j}$ to include a caller i in FCS thus degrading the lookup performance. However, if it is set too low, then callers who do not call j often will join FCS degrading the move performance.

In order to compute $LCMR_{i,j}$, each caller i maintains $C_{i,j}$ and sends it to j with every call. Similarly each mobile user j maintains M_j and sends it to the new visited VLR at every move. Maintaining $C_{i,j}$ and M_j involves collecting these values over a time period T. A simple

method is chosen which clears $C_{i,j}$ and M_j (i.e. set them to zero) at the start of each interval. In order to avoid transient effects in each period, the values computed during the period T_i are the ones used while computing the new values during the period T_{i+1}. This excludes the special case of T_0, i.e. when the user joins the system for the first time. In this case, the values used are the ones computed during T_0 which might reflect transient values. If the value of T is set too large, it will result in not capturing the dynamic behavior of moves and calls, therefore decisions would be based on long ago activities. However, setting T to a very small value will suffer from transient effects. This simple strategy for maintaining $C_{i,j}$ and M_j is one of many that have been proposed previously [5,9,15,16].

Consider a vector V_j containing $C_{i,j}$ for all i that sent calls to j. V_j is maintained and kept by the VLR hosting the mobile. When a new call arrives from caller k, then the corresponding entry $C_{k,j}$ in V_j is updated. Each entry in V_j is associated with a timestamp to serve as an indicator of how valid the value of $C_{i,j}$ is. Each mobile user j is associated with a profile P_j ={FCS, V_j}. When mobile j moves to a new VLR, the old VLR passes the profile P_j to the new VLR during the handover procedure. The values of V_j are then used by the VLR to compute LCMR and determine whether a caller can be classified as a frequent caller. A list of the symbols used can be found in table 1.

The members of FCS are updated in two cases. The first case is when a new call is received. Consider a mobile user j, hosted by VLR_x, receiving a call from a caller i. If i does not belong to FCS, then VLR_x computes $LCMR_{i,j} = C_{i,j} / M_j$. If $LCMR_{i,j} > \beta$ then i is added to FCS. The second case is when j moves from location VLR_x to VLR_y. Since the location of the mobile has changed as well as M_j, then VLR_y needs to reevaluate $LCMR_{i,j} > \beta$ for all callers with $C_{i,j} \neq 0$ and reconstruct the FCS. However, for high mobility the FCS is reevaluated only at the K^{th} move. This is due to when using forwarding pointers the members of the FCS are updated only when purging the chain. It should be noticed that during the reevaluation of the FCS, the members which are not classified anymore as frequent callers must be invalidated such that subsequent calls from those members are forced to search through the HLR.

In deciding on the on-line threshold β, we consider the total database and communication cost associated with lookups and moves. It is beneficial to include a caller i in FCS if the gain in the lookup is more than the extra cost paid in updating and invalidating i when the mobile j moves. Let $D_{i,HLR}$ be the distance between the caller i and the HLR of j, $D_{j,HLR}$ the distance between j and HLR of j and $D_{i,j}$ the distance between the mobile j and the caller i. Further, let C, the communication cost of sending a message one hop away, U, the cost of updating a database and Q, the cost of querying a database. The cost of issuing a query from one node to another consists of the cost of querying the database (Q) and the communication cost of sending the message and receiving its acknowledgment (2*C*distance between the two nodes). Similarly, the cost of issuing an update from one node to another involves the cost of updating the database (U) and the communication cost of sending the message and receiving its acknowledgment (2*C*distance between the two nodes). Then it is beneficial to include i in FCS if $C_{i,j}\ [2Q + 2C(D_{i,HLR} + D_{j,HLR}) - (Q + 2C*D_{i,j})] \geq [U + 2C * D_{i,j}]\ M_j$. That is,

$$\beta \geq \frac{U + 2C * D_{i,j}}{Q + 2C(D_{i,HLR} + D_{j,HLR} - D_{i,j})} . \qquad (1)$$

The values of U,Q and C can be set to values reflecting the prominence of the database versus the communication cost. For instance by setting Q=U=0, the classification will be done with emphasis on the communication cost. It is observed from eq. (1) that as the caller and the callee are close to each other and away from the callee's HLR, this will result in smaller values of β, favoring the inclusion of the caller in the FCS to reduce the lookup cost. As the distance between the caller and the callee increases relative to their distance to the callee's HLR, the value of β increases reducing the possibility of the caller joining the FCS. The distance between two nodes could be represented by the number of hops between them. The distances $D_{i,j}$ and

$D_{j,HLR}$ can be determined using the IP time-to-live (TTL field) of messages received by the VLR from the caller and the HLR respectively. We assume that each VLR has an internal routing table that keeps the distances between various callers and the HLR of mobiles. It is expected that the distance between the callers and the HLR of mobiles will not change very often. Thus, once the routing table is computed, it can be maintained in memory without much effort.

3. Performance analysis

In this section, we present a preliminary analysis of the potential benefits of the adaptive scheme using some simplifying assumptions. Let $D_{HLR,VLR1}$ be the distance between the HLR and the first VLR in the chain of forwarding pointers, $D_{i,VLR1}$ be the distance between the caller and the first VLR in the chain, η the fraction of calls received from callers belonging to the frequent caller set and N_s the number of callers in FCS.

Let the cost of the Basic Move denoted by M, the cost of Basic Lookup by L, the cost of the Adaptive Move by M' and the cost of Adaptive Lookup by L'. Let the CMR denoted by p, the total cost of the Basic Scheme be C_B and the total cost of the Adaptive Scheme be C_A.

Table 1
List of symbol description

Symbol	Description
$C_{i,j}$	Number of calls made by caller i to mobile j in a time period T
M_j	Number of moves made by mobile j in a time period T
CMR	Call to Mobility Ratio = $\Sigma_{\forall i} C_{i,j}/M_j$
LCMR	Local Call to Mobility Ratio = $C_{i,j}/ M_j$
C	Communication cost of sending a message one hop away
U	Cost of updating a database
Q	Cost of querying a database
K	Maximum length of forwarding pointer chain
V_j	Vector containing $C_{i,j}$ for all i that sent calls to mobile j
$D_{i,HLR}$	Distance between caller i and HLR of j
$D_{j,HLR}$	Distance between mobile j and its HLR
$D_{i,j}$	Distance between caller i and mobile j
$D_{HLR,VLR1}$	Distance between HLR of j and the first VLR in the chain of forwarding pointers
$D_{i,VLR1}$	Distance between caller i and first VLR in the chain of forwarding pointers
Ns	Number of callers in FCS
η	Fraction of calls received from FCS
α	Threshold separating between high and low mobility
β	Threshold separating between frequent and non frequent callers

Then,

$C_B = M + pL$

$C_A = M' + pL'$.

The cost of the Basic Lookup consists of the cost of querying the HLR (i.e. Q + 2C* $D_{i,HLR}$), then querying the VLR (i.e. Q + 2C *$D_{j,HLR}$). That is,

$L = (Q + 2C^* D_{i,HLR}) + (Q + 2C * D_{j,HLR})$. (2)

The cost of Adaptive Lookup depends on whether the caller i belongs to the FCS or not and whether forwarding pointers are used or not. For simplicity, we assume that the forwarding pointers are placed one hop away from each other, that is the distance between two consecutive VLRs in the trajectory of the user is one. In case of using forwarding pointer ($p < \alpha$) , if the caller i is not a frequent caller, then the lookup cost consists of querying the HLR ($Q+2*C*D_{i,HLR}$), then querying the first VLR in the chain of forwarding pointers ($Q+2*C*D_{HLR,VLR1}$), then following the chain of pointers till j is found ($K*Q+2*C*K$). In case i belong to FCS, the cost consists of querying the first VLR in the chain ($Q+2*C*D_{i,VLR1}$), then following the chain of pointers ($K*Q+2*C*K$). In case forwarding pointers are not used ($p \geq \alpha$), then the lookup cost is represented by eq. (3-a) which is obtained by setting K to 0 in eq. (3-b) taking into consideration that in this case VLR_1 – the first VLR in the chain - and j are the same.

$$L' = \begin{cases} \eta[Q + 2C * D_{i,j}] + (1-\eta)[2Q + 2C * (D_{i,HLR} + D_{j,HLR})] & p > \alpha, \\ \eta[Q + 2C * D_{i,VLR1} + K(Q+2C)] + (1-\eta)[2Q + 2C * (D_{i,HLR} + D_{HLR,VLR1}) + K(Q+2C)] & p < \alpha, \end{cases} \quad (3\text{-a}), (3\text{-b})$$

Consider a move operation where a user j moves from location x to location y. For the Basic Scheme, the cost of Basic Move consists of the cost of updating HLR (i.e. $U+2*C*D_{j,HLR}$), registering j at VLR_y (U) and deregistering j at VLR_x (i.e. U+2*C).

$M = 3U + 2C(D_{j,HLR} + 1)$. (4)

For the Adaptive Scheme, the cost of Adaptive Move depends on whether forwarding pointers are used or not. In case $p \geq \alpha$, the cost is the same as the basic scheme in addition to the cost of updating the set of frequent callers in FCS. For $p < \alpha$ the cost consists of updating VLR_y (U) and setting a forwarding pointer at VLR_x (U+2*C). Further, at the K^{th} move, the cost includes invalidating the chain $K*(U+2*C)$, updating HLR of j ($U+2*C*D_{j,HLR}$) and the cost of updating the set of callers in FCS ($N_s*U+2*C*\Sigma_{\forall i \in FCS} D_{i,j}$).

$$M' = \begin{cases} M + NsU + 2C * \sum_{\forall i \in FCS} Di, j & p \geq \alpha \\ (1/K)[U(K + Ns + 1) + 2C(K + Dj, HLR + \sum_{\forall i \in FCS} Di, j)] + (1 - 1/K)(2U + 2C) & p < \alpha \end{cases} \quad (5)$$

4. Results and analysis

The above equations can be used to estimate the benefits and costs of the proposed strategy. To illustrate the effect of the communication cost, Q and U are set to zero and C is set to 1. In the following, we plot C_A/C_B denoting the relative total cost of the proposed strategy versus the Basic strategy over a range of Call to Mobility Ratios ranging between 0 and 4. The various parameters are set as follows. The value of α, which is the threshold separating between low and high mobility, is set to 2 [10]. The distance $D_{i,j}$ is set to 2, $D_{i,VLR1}$ is set to 5 and $D_{i,HLR}$, $D_{j,HLR}$ and $D_{HLR,VLR1}$ are set to 10. This setting represents a case where the MH is closer to the frequent caller set than to its HLR. We conduct several experiments to study the effect of the various parameters on the performance of our algorithm.

The effect of varying η, the fraction of calls generated from the set of frequent callers, is shown in fig. 1. This effect is studied for K=20, N_s= 3 and η is set to 0.2, 0.4, 0.6, 0.8 and 1. We notice that, for CMR < 0.5, the update cost is more dominant and the saving in update messages caused by the use of forwarding pointers lead to improvement in the total cost. As η increases, the saving in the total cost increases from 30% to 50% for different values of η thus leading to better performance. This is due to the decrease in the lookup cost since

most calls are routed directly to the MH. As the CMR increases, the effect of the lookup cost begins to outweigh the saving in the update cost, due to the presence of forwarding pointers. Thus the saving in the total cost decreases and approaches the basic scheme especially for smaller values of η. For CMR $\geq$ 2, an improvement in the performance is observed, especially for large values of η. This is due to the fact that the algorithm switches to the informed scheme and ceases using forwarding pointers for high CMR leading to a better lookup cost. As CMR increases, the improvement in the total cost increases reaching up to 60% for large values of η. For small values of η, the improvement is minimal (9% for η = 0.2) since most calls follow the basic scheme.

Fig. 2 shows the effect of varying the length of the forwarding pointer chain, K, on the total cost for different values of CMR < 2. The value of η is set to 0.8 and N_s is set to 3. The scheme is evaluated for K=2, 6, 10, 20, 30 and 40. We can observe that, as the chain length increases, the total cost of our proposed strategy increases due to the excess in the lookup cost. The increase is higher for larger values of CMR since the lookup cost becomes more dominant. For very small values of K (K=2), the improvement in cost is small (20%) for very low values of CMR since the updates are performed every second move, thus reducing the savings of the scheme but it reaches 60% for large values of CMR. For K=10, the improvement varies from 43% to 60%. For K=20, it ranges from 25% to 57%. However, when K=30 and 40, the cost increases and approaches, or even bypasses, the basic scheme especially for high values of CMR. It is deduced from this experiment that choosing the value of K highly affects the performance with very small values or very large values leading to clear degradation. For CMR $\geq$ 2, an improvement ranging between 35% to 46% has been observed irrespective of K since forwarding pointers are not used.

The effect of varying the size of the frequent caller set, N_s, for K=20 and η=0.8 is considered. Figs. 3 and 4 show the performance for CMR < 2 and CMR $\geq$ 2 respectively for N_s set to 2, 4, 6, 10 and 15. For low CMR, the improvement in the update cost decreases as N_s increases. This is expected as larger size of the FCS causes higher number of update messages sent by the MH. However, the variation in the cost is minor for the set size ranging from 2 to 15 since update messages are only sent when purging the forwarding pointer chain. As CMR increases, the improvement in the total cost decreases from 60% to 25% due to the excess in the lookup cost caused by following the forwarding pointer chain. To illustrate the benefits gained from using the forwarding pointers for low CMR compared to the informed scheme, the corresponding case for N_s=6 is plotted. As observed, the improvement of our strategy outperforms the informed scheme especially for low CMR.

Fig. 4 shows the effect of varying N_s for CMR $\geq$ 2. As shown, increasing the set size reduces the improvement in the total cost from 50% to 17% due to higher number of updates. For $N_s \geq 15$, the proposed strategy becomes almost as the basic strategy. However, it is observed that, as CMR increases, the gain improves reaching 28% to 58% for high CMR due to the enhancement in the lookup cost, which dominates.

The above experiments were conducted for the case where the MH is closer to its callers than to its HLR. Other cases representing different variations of the MH position relative to its HLR and to its callers can be studied by varying the values of the set of distances chosen, that is $D_{i,HLR}$, $D_{j,HLR}$, $D_{i,j}$, $D_{HLR,VLR1}$ and $D_{i,VLR1}$. In fig. 5, we evaluate the cost and benefits of some of those variations for η = 0.8, N_s = 3 and K=10 for different values of CMR. We have considered 4 cases as follows. Case 1 represents all entities close to each other. In this case, the improvement of our strategy is about 15% for low values of CMR and decreases as CMR increases approaching the basic scheme. Degradation over the basic scheme of 10% is observed for 1.4<CMR<1.8. This degradation is due to the fact that the benefit of using forwarding pointers is minimal when the distances are small. For CMR>2 an improvement is observed which increases as CMR increases to reach 22% for large CMR.

Case 2 represents the situation where the MH is away from its HLR and from its callers while the callers are close to the MH's HLR. An

improvement over the basic scheme is observed for CMR<2 ranging from 20% to 40%. As CMR>2, a degradation ranging between 40% to 70% can be seen. This degradation is caused by the long distance between the MH and its callers, which must be informed after each move thus increasing the update cost.

The situation where the MH is close to its HLR but both are away from the callers is studied in case 3. We can notice degradation in performance ranging from 27% to 35% over the basic scheme for CMR<2. The degradation increases as CMR reaches the threshold α. This is due to the large overhead incurred when informing the set of frequent callers, which are away from the MH, after each move. As CMR gets larger, the degradation improves, but still the basic scheme outperforms the proposed algorithm in this case.

Finally, in case 4 we studied the situation where all entities are away from each other. In this case, the use of forwarding pointers for low CMR proves to enhance the performance by 40% for small CMR. This improvement decreases as CMR increases reaching 22%. For CMR > 2, a degradation in performance appears, around 25%, that decreases as CMR increases until approaching the basic scheme for large values of CMR.

As observed from these experiments, the proposed algorithm outperforms the basic scheme in most cases, approaching it in some cases and inferior to it in only few cases.

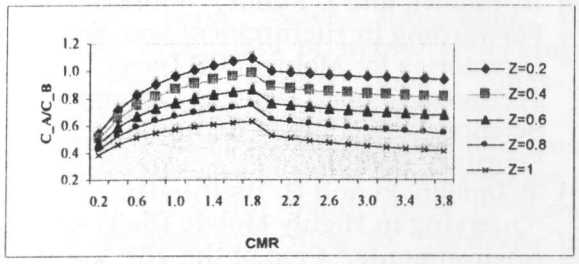

Fig 1. Varying the fraction of calls initiated from the FCS.

5. Conclusions and future work

Locating users in mobile environment is an essential problem in PCS. In this paper we present an adaptive location strategy using forwarding pointers that reduces the overall location management cost. It is driven by the

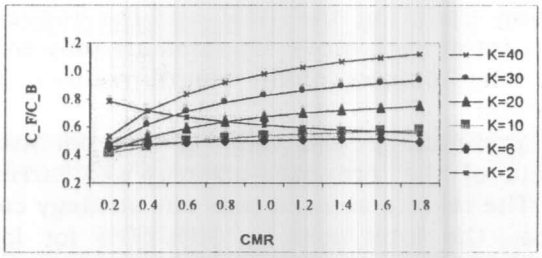

Fig. 2. Varying the pointer chain length.

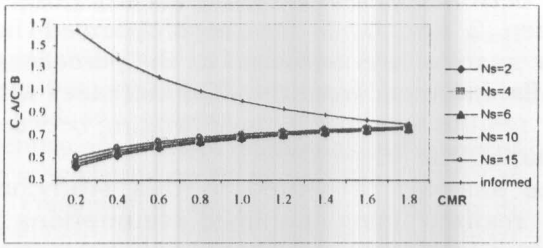

Fig. 3. Varying size of FCS for low CMR.

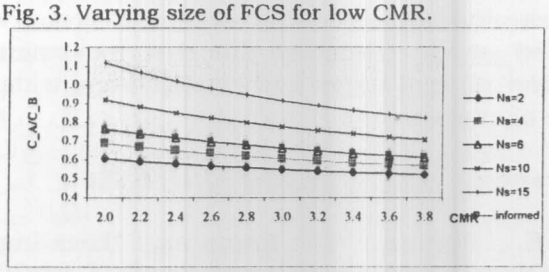

Fig. 4. Varying size of FCS for large CMR.

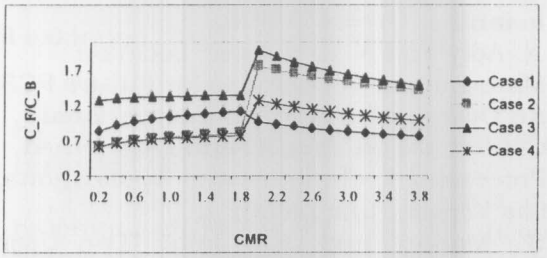

Fig. 5 Varying the set of distances.

observation that the set of callers that initiate most of the calls for a mobile host is small and stationary. Thus, informing this set would reduce the lookup cost but at the expense of increasing the update cost. This technique is efficient for high Call-to-Mobility Ratio. However, the increase in the update cost is significant for low CMR, especially in third generation PCS where update signaling is expected to increase dramatically. Therefore, the proposed scheme is augmented by using forwarding pointers for low CMR. The use of forwarding pointer reduces the update cost by

avoiding updating the HLR and the frequent caller set at each move. Updates are only sent when the pointer chain length reaches its maximum.

A preliminary analysis of the potential benefits of the proposed strategy was carried out. The results showed that our strategy can reduce the total cost by 30%-60% for low CMR, and by 20%-50% for high CMR. These results are achieved for a pointer chain length lying between 10 and 20 and FCS size ranging between 2 and 6. It has been observed that better results are achieved as the percentage of calls initiated from the FCS increases with good results for a percentage ranging between 60% and 80%.

The analysis presented in this study has been realized using simplified assumptions to obtain a first estimate. It would be interesting to examine the results obtained from a detailed study modeling different movement patterns of real users and conducting a wider range of experiments.

References

[1] E. Pitoura, G. Samaras, "Locating Objects in Mobile Computing," *TR 98-20*, Computer Science Dept., University of Ioannina, Greece (1998).

[2] N. Adly and A. El Nahas, "Location Management Techniques for Future PCS: An Overview", The third International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, June (2000).

[3] S. Mohan and R. Jain, "Two User Location Strategies for Personal Communication Services," IEEE Personal Communications, Vol. 1(1), pp. 42-50, January (1994).

[4] D. B. Johnson and D. A. Maltz, "Protocols for Adaptive Wireless and Mobile Networking," IEEE Personal Communications, Vol. 3(1) (1996).

[5] S. Rajagopalan and B. Badrinath, "An Adaptive Location Management Strategy for Mobile IP", ACM MOBICOM pp. 170-180 (1995).

[6] R. Yates, C. Rose, S. Rajagopalan and B. Badrinath, "Analysis of a Mobile-Assisted Adaptive Location Management Strategy", ACM Mobile Networks and Applications (1996).

[7] N. Shirakumar, J. Widom, "User Profile Replication for Faster Location Lookup in Mobile Environments," Proc of 1st ACM Intl. Conf. On Mobile Computing and Networking, Mobicomm 95 (1995)

[8] J. Jannick, D. Lam, N. Shivakumar, J. Widom and D. Cox, "Efficient and Flexible Location Management Techniques for Wireless Communication Systems", Proc of 1st ACM Intl. Conf. On Mobile Computing and Networking, Mobicomm 96 (1996).

[9] D. Lam, Y. Cui, D. Cox and J. Widom, "A Location Management Technique to Support Lifelong Numbering in Personal Communications Services," Proc of Global Telecomm Conference (GlobeComm 97), Vol 2, pp. 704-710 (1997).

[10] R. Jain and Y.B. Lin, "An Auxiliary User Location Strategy Employing Forwarding Pointers to Reduce Network Impact of PCS," ACM Baltzer J. Wireless Networks, Vol.1, pp.197-210 (1995).

[11] Y. B. Lin, "Location Tracking with Distributed HLRs and Pointer Forwarding", IEEE Transaction on Vehicle Technology, Vol. 47(1), pp. 58-64 (1998).

[12] E. Pitoura and I. Fudos, "Tracking Mobile Users using Hierarchical Location Databases", Proc of the 6th Panhellinic Conference on Informatics, Nov. (1997).

[13] E. Pitoura and I. Fudos, "Address Forwarding in Hierarchical Location Directories for Mobile PCS Users," Technical Report, Dept. of Computer Science, University of Ioannina, February (1998).

[14] T. Imielinski and B. Badrinath, "Querying in Highly Mobile Distributed Environments," Proc of the 18th VLDB Conference, Canada (1992)

[15] S. Tabbane, "An Alternative Strategy for Location Tracking," IEEE Journal on Selected Areas in Commun., Vol. 13, pp. 880-892 (1995).

[16] E. Pitoura and G. Samaras, Data Management For Mobile Computing, Kluwer Academic Publishers (1998).

Received Augusts 10, 2000
Accepted April 20, 2001