A context-aware cache structure for mobile computing environments

Stylianos Drakatos a,*, Niki Pissinou a, Kia Makki a, Christos Douligeris b

a Telecommunication and Information Technology Institute, Department of Electrical and Computer Engineering, Florida International University Miami, FL 33174, USA
b Department of Informatics, University of Piraeus, Piraeus 18534, Greece

Received 14 April 2006; received in revised form 5 October 2006; accepted 10 October 2006
Available online 28 November 2006

Abstract

This paper proposes a cache management method that maintains a mobile terminal’s cache content by prefetching data items with maximum benefit and evicting cache data entries with minimum benefit. The data item benefit is evaluated based on the user’s query context which is defined as a set of constraints (predicates) that define both the movement pattern and the information context requested by the mobile user. A context-aware cache is formed and maintained using a set of neighboring locations (called the prime list) that are restricted by the validity of the data fetched from the server. Simulation results show that the proposed strategy, using different levels of granularity, can greatly improve system performance in terms of the cache hit ratio.

Published by Elsevier Inc.

Keywords: Context-aware information service; Mobile computing; Location-awareness; Caching; Prefetching

1. Introduction

Data caching as a means to achieve higher performance has been a perpetual quest in the computer industry. Most recently and due to the significantly increased size of memory available for small devices, caches are used as small data base systems that hold content most likely to be used in the near future (this is called prefetching or hoarding). However, due to the size limitation of the cache, cache management strategies exist to efficiently manage the cache content. The location information is used as a key field of the user’s query context, but not enough attention has been given to the other query fields (predicates) which define the user’s information context. Emerging location-based service (LBS) providers use the location information of mobile users to provide them with relevant information based on their geographical positions. Information disseminated to mobile users potentially can be context-sensitive and highly personalized. Therefore, an effective cache management scheme needs to adapt dynamically to the user’s query context. Additionally, both the cached data items and the prefetched ones should be determined and adjusted according to the user’s movement pattern and information context.

In evaluating the data item’s benefit as far as the cache content is concerned, we propose a scheme that uses the query context as an information filtering mechanism to limit the amount of prefetched information to the data items with maximum benefit. A main aspect of this work involves predicting the future context that will be required by the user. In some situations forecasting may be impossible, but in situations where the content is changing gradually and continuously i.e., in continuous type of queries, this may be possible and very effective. Forecasting may be done, for example, by analyzing the user’s current query context. The purpose of trying to predict future contexts is to anticipate the user’s future retrieval needs, and to perform retrievals in advance of the need. Assuming the
prediction is correct, the response to retrieval requests will then be very fast, since the necessary retrieval will have been done in advance. When a cache-miss happens, the mobile terminal (MT) asks for several other items and not just the cache-missed data item, with little additional cost. This action will prevent future cache misses and will reduce the number of uplink requests.

A mobility-based semantic cache structure and query processing was first proposed in Dar et al. (1996). Ren and Dunham (2000) and Ren et al. (2003) have extended this work to use the location information attached to each segment, making it more efficient and they have also proposed a cache management replacement policy. This work is based on this previous body of research on semantic cache management, however it focuses on the cache management prefetching strategy. To design an effective cache management strategy we consider the neighboring cells with valid information and the current query information context factors. Based on these two factors, we outline the major contributions of this paper below:

- The key contribution of this study is a context-aware (CA) cache management prefetching strategy, which first uses the validity of the data (valid scope distribution) based on their location to derive a set of most likely future cells called the “prime” list of cells. Next, in order to identify data items with high benefit as far as the cache content is concerned the user’s query context is exploited to limit the amount of prefetched information within the predicted set of future cells (the prefetching zone).
- A direct result of the proposed strategy is the formation and maintenance of the context-aware cache of data items with a high cache value which are included at a low cost. The context-aware cache is then updated if the mobile user subsequently strays out of the predefined prime list of cells.
- The performance of the cache management strategy is examined through a number of experiments which show that the context-aware prefetching produces significant improvements over the standard direction velocity based prefetching strategies.

The rest of the paper is organized as follows: Section 2 gives a brief description of location-dependent data caching and prefetching strategies-related research. Section 3 presents the mobile architecture together with the mobility and query models. This section also explains the semantic cache description at the mobile client. Section 4 presents the proposed cache management strategy and explains its associated components, i.e. the presentation of the query-context prefetching method and prefetching algorithm. Section 5 presents a prefetching cost analysis. Section 6 describes the simulation model and performance comparison is discussed in Section 7. Finally, Section 8 provides conclusive remarks and future plans regarding the presented research.

2. Previous work

Research on cache management has been active over the past few years. The least recently used (LRU) replacement policy, which evicts the object that has not been accessed for the longest time, works well when the most recently referenced objects are most likely to be referenced again in the near future. Ren and Dunham (2000) have proposed a mobility model to represent the moving behavior of mobile users and have formally defined location-dependent queries. Based on their mobility model, they developed a Location Dependent Data (LDD) semantic caching scheme called Furthest Away Replacement (FAR), which implies future location prediction based on tangent velocity. A body of previous research (Lee et al., 2002; Zheng and Lee, 2002; Xu et al., 2003), incorporated the valid scope as an important factor in data caching replacement policy using the geographical mobility models. Akyildiz et al. (1996) and Levine et al. (1997) proposed the use of the shadow cluster model for future location prediction and used it for resource allocation in asynchronous transfer mode (ATM)-based wireless networks. Akyildiz and Wang (2004) have enhanced their approach by including user profiling. They proposed a matrix of transition probabilities using the shadow cluster model, considering all possible cell locations, for a total of $3k^2 + 3k + 1$ cells, each assigned to one state, where $k$ is the number of rings. However, this number of states explodes quickly as $k$ increases, sometimes making analysis and simulation difficult and costly. A good amount of redundancy is built into this approach because not all of these cells are of interest if they have no valid answers for the query. Mao and Douligeris (2000) have used the random movement model with cell location granularity in their location-based mobility management research.

In Liu et al. (1998), a two-level user mobility model is used to represent the movement behavior at global and local levels. The next cell is predicted by considering speed and direction of a user’s trajectory. Through estimation of mobile users’ trajectory and arrival/departure times in Aljadhai and Znati (2001), a group of future cells are determined, which constitute the most likely cluster into which a terminal will move. Most recently, Park et al. (2004) have proposed prefetching policies based on the current position and the velocity of the MT and used the value of the tangent velocity to predict the future location of the MT. This method uses the geographical mobility models and is effective only within a short time interval $\Delta t$. In order to limit the amount of prefetched information, current research has used geographical mobility models to focus solely on the mobile user’s movement pattern (Ren and Dunham, 2000; Duhham and Kumar, 1998; Zheng and Lee, 2002; Akyildiz et al., 1996). The geographical mobility models inherently use continuous calculations of the tangent velocity, which is proven to have considerable high processing overhead (Satyanarayana, 2002; Datta et al., 2003). In summary, most of the existing methods are aimed at
finding the most probable cell (Aljadhai and Znati, 2001). However, when an MT moves quickly in micro-cell networks, the short residence time in a cell may not allow computations in every cell, i.e. next-cell prediction.

3. Mobile architecture

In a mobile architecture (Fig. 1), the geographical coverage area for the information service is partitioned into service areas, with each service area attached to a data server. The service area may cover one or multiple cells. Each service area is associated with a service_id for identification purposes. This id is broadcasted periodically to all the mobile clients in that service area. The database associated with each service area is a collection of data items. Every data server keeps a complete copy of the database, i.e., the same data items are replicated on all the data servers but probably with different values in different data servers. MTs and the fixed data servers can communicate with each other through wireless channels via Mobile Switching Stations (MSSs). MSSs are the elements that control several base stations (BSs) and have the capability to execute software that controls communications among wireless devices, providing them with access to the wired network. MSSs exchange control and user location information. This kind of network architecture, with the presence of MSSs, is known as an infrastructure network. In most cases, the remote server and the cells (e.g. base stations) are connected through wireline links, while the wireless link is used only for the last hop between the MT and the base station.

Prefetching takes place between the remote server and the base stations i.e., through wireline links, or even with high bandwidth wireless links. However, due to the dramatic increase in memory size of the mobile devices, caches commonly located at proxy servers may now be co-located with the clients and used for prefetching. This alternative method of prefetching through wireless links happens between the MT and the remote database server. The proposed prefetching strategy proactively loads information from a remote database server associated with a reduced number of future cells and uses a filtering mechanism which explores the current query context.

3.1. Modeling user mobility

Various mobility models may be used to analyze and emulate the behavior of mobile users in the mobile environment. The random walk (RW) mobility model is more suitable for personal communications applications where most of the subscribers are likely to be pedestrians (Lam et al., 1997; Madhow et al., 1995). In this work we primarily use the RW model and as a secondary model we have included the directional (DIR) movement model. However, our model and pre-fetching technique is independent of the mobility model we select in the experimental section. The motion models we consider (analogous to motion models already presented in previous works Akyildiz et al., 1996; Levine et al., 1997; Akyildiz and Wang, 2004; Mao and Douligeris, 2000) serve just as examples of practical application, other models could be adopted as well.

We consider motion in an area divided into adjacent “cells” (Fig. 2). Each cell consists of all the locations that share common information that must be provided by the service as answer to a user query, when the user stays within those cells. The motion model is defined as follows: At the end of each time slot, the user can remain in the same cell, or move to an adjacent cell through one of the shared edges. It is assumed that a MT resides in a cell for a generally distributed time interval before it moves on to one of the adjacent cells with a uniform probability of $\frac{1}{6}$. The user moves to an adjacent cell with probability $c$, or remains in the same cell with probability $\frac{1}{6} - c$. Given this motion model, a natural prefetching strategy is to prefetch information concerning cells that are within a given “radius” from the current position. To this purpose,
we define as distance between two cells the minimum number of cells that must be traversed to pass from one cell to the other one, and as ring $k$ the set of all cells whose distance from a given cell is equal to $k$. The prefetching-strategy outlined above prefetches all the information associated to rings $0, 1, 2, \ldots, k$ around the starting position, for a given $k$ (circle of radius $k$). No remote loading is needed until the user moves within circle $k$. When the user enters a cell outside of a cluster of $k$ rings, that cell becomes the new starting position and a new cluster of size $k$ is reconstructed around this location by loading the needed information from the remote data server (note that some of this information is already loaded by the previous prefetching).

### 3.1.1. Location granularity

Being discrete and well structured, location information based on symbolic location models is easier to manage compared to that based on geometrical models. For example, location data is much more amenable for database storage and retrieval; they can help analyze location information such as individual mobility patterns. Steadily falling costs of storage lead to caches of sizes large enough to hold most of the additional requested data items by the proposed prefetching technique. Additionally, the fact that a cache can easily incorporate the functionality of a small database,\(^1\) can provide additional motivation for the use of the symbolic location model based on cell granularity. The symbolic model stresses the representation of relationships between logical entities rather than their precise coordinates and is more suitable for LBS at a semantic level (Xu et al., 2003). Additionally, cell-based location identification requires neither additional devices deployed on mobile clients nor modifications over the current cellular network infrastructure. Thus, this is the cheapest solution (Lee et al., 2002). With recent developments in micro-cell, pico-cell and nano-cell systems\(^2\) it is believed that for most of the emerging mobile LBS applications the cell id number will be the preferable granularity (Zheng and Lee, 2002).

Determining the exact location requires satellite technology which is still not widely available in cell phone networks in many countries. Cellular communications themselves have spawned the concept of assisted GPS (AGPS) where the network assists the GPS receiver to perform its various functions. The most demanding AGPS environments tend to be in inner-cities where cell sizes can be limited to a few kilometers in radius (www.gps-world.com). The technology to perform mobile positioning (MP) under WAP is proposed, which can apply to the existing GSM/DCS networks without doing any modification to the GSM/DCS standard.

Cell ID is a simple method for mobile phone positioning and is widely commercially deployed. The technique determines the location according to the strongest base station signal and end-device receives and thus the approximate position of the user that uses the cell area (or Cell ID) of the caller. The NTT-DoCoMo and J-phone in Japan have been using the cell granularity (Cell ID) techniques to provide basic LBS applications since 1999. The location sensing applications convert the Cell ID into a symbolic location, and presenting it to different instant messaging (IM) networks. A key feature of this type of application is the ability for the users themselves to define new places as a combination of the current Cell ID and some semantic information describing the place. A pro-active “friends finder” location sensing application provided by AT&T in the US and TeliaSonera in Sweden, creates services based on cell granularity.

Recent research on the emerging mobile multimedia applications QoS-based efficient Resource provisioning has been also based on the current location of mobile clients at the cell level. For example “Information collection services for QoS-aware mobile applications (2003)” by Qi Han and Nalini Venkatasubramanian, Department of Computer Science, University of California at Irvine. Collecting mobility information at a finer level (i.e., for each individual mobile client) would facilitate adaptive resource provisioning; however, it would incur significant overhead without extreme performance improvement.

### 3.2. The client’s semantic cache description

In this section we give a brief presentation of the semantic cache using a logical model (Fig. 3) that uses the cell granularity for location identification instead of the $(x,y)$ geographical mobility models used by previous researchers

---

\(^1\) True relational DBMS, may now fit into 100K–150 K of memory (e.g., DB2 Everywhere by IBM, Oracle Lite and Sybase’s Ultra Lite databases).

\(^2\) The average cell size in diameter, in a typical micro-cell system, is 100 m–1 km, in a typical pico-cell system is 10–100 m in a typical nano-cell system, is 1–10 m in diameter (Taylor et al., 1997).
The mobile client’s semantic cache stores additional information (metadata) such as query results (the data items component) and the query descriptions (the index component) which is consistent with the definition of the LDD query (Ren and Dunham, 2000). The metadata is used to determine whether a new query is fully answerable using the cache contents, in which case no communication with the server is required. If the query can be only partially answered, then it is trimmed and sent to the server (remote query). The query part that is satisfied by information already in the cache is called the local (explore) query. Another form of cached metadata is the location binding information that is used for both replacement and prefetching. The execution of the query Q will bring values that will be maintained in the storage cache as a Semantic Segment, which can be defined as a tuple arrangement $S = (S_R, S_A, S_p, S_u, S_L, S_m)$.

In this definition, $S_R$ and $S_A$ are respectively the base relation and the attributes in $S$. $S_p$ represents the link to the first page that stores the segment. $S_a$ is the timescale indicating when the segment was last accessed by the cache manager. $S_p = P_1 \lor P_2 \lor \cdots \lor P_m$ indicates the criteria which the tuples in the semantic segment $S$ satisfy. In the $S_p$ equation, $P_i$ is a conjunction of simple predicates, i.e., $P_j = b_{j1} \land b_{j2} \land \cdots \land b_{ji}$. Each $b_{ji}$, where $i = 1, 2, \ldots, I$ is a simple predicate. $I$ is the project operation that lists the subset $(\Pi_{x_j} S)$ of attributes defined by the query and $\sigma$ is the select operation, that selects the tuples $(\sigma_{x_j} S)$ to satisfy the predicates requested by the query.

Example 3.2 (Continuous query-cache state). Consider a yellow pages relational database, where the mobile user asks the following questions to find the nearest restaurant, the results of which are cached afterwards. At time $t_1$ and location $cell_id = x_1$, the mobile user asks query $Q_1$: “Give me all the names of the nearest hotels (within 5 miles)”, and then keeps on driving to the next location. The results of $Q_1$ are cached as segment $S_1$. Then at time $t_2$ and location $cell_id = x_2$, s/he asks query $Q_2$: “Give me the names and types of the restaurants within 10 miles that are open from 6:00 p.m. to 9:00 p.m.”, and then keeps on driving to the next location. The result of $Q_2$ is cached as segment $S_2$. Finally at time $t_3$ and location $cell_id = x_3$, s/he asks query $Q_3$: “Give me the names and vacancy information for hotels within 5 miles which charge up to $100$”. The result of $Q_3$ is cached as segment $S_3$. Assume that the first pages of $S_1$, $S_2$, and $S_3$ are 2, 5 and 8 respectively; a snapshot of the cache state with the three cached segments is shown in Table 1.

<table>
<thead>
<tr>
<th>$Q_i$</th>
<th>$S_i$</th>
<th>$S_A$</th>
<th>$S_p$</th>
<th>$S_u$</th>
<th>$S_L$</th>
<th>$S_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_1$</td>
<td>$S_1$</td>
<td>Hname</td>
<td>$x_1$</td>
<td>$t_1$</td>
<td>$x_1$</td>
<td>2</td>
</tr>
<tr>
<td>$Q_2$</td>
<td>$S_2$</td>
<td>Rname type</td>
<td>$x_2 \land (6 \leq \text{schedule} \leq 9)$</td>
<td>$t_2$</td>
<td>$x_2$</td>
<td>5</td>
</tr>
<tr>
<td>$Q_3$</td>
<td>$S_3$</td>
<td>Hname vacancy</td>
<td>$x_3 \land (\text{price} \leq 100)$</td>
<td>$t_3$</td>
<td>$x_3$</td>
<td>8</td>
</tr>
</tbody>
</table>

4. The cache prefetching components

Prefetching is a technique that is mainly concerned with improving the system performance. Caching alone is generally not enough to improve performance of mobile systems. Moreover, prefetching has a broader application range than simply storing already used data in a cache. Prefetching together with replacement are used to support cache management. To avoid excessive network traffic and prefetching cycles, the prefetching mechanism has to consider different strategies to increase the efficiency of the algorithm and the relevance of the fetched data. The prefetching mechanism may take any of the following filter parameters into account.

- **Movement pattern**: A user’s movement pattern (i.e., location and direction).
- **Query pattern**: The priority of services defined by the user (query pattern).
- **User’s profile**: A user’s interests through user profiling.

The continuous type of query this study focuses on, shows an affinity towards certain data items, which are called query patterns. A query pattern is a parameter which together with the user’s movement pattern is what make our prefetching strategy unique compared to other proposals. The third parameter deals with the user’s profiling which we have chosen to be a future direction of this research. Next in this section, the valid scope concept is presented, followed by the movement and query pattern factors of the proposed scheme.

4.1. The valid scope concept

A cell is defined as a limited geographical area where a base station covers a number of mobile clients. Each cell is surrounded by rings of cells and the innermost cell is considered to be the center cell for analysis purposes. Combinations of cells where the data item value has valid answers is defined to be the data item value valid scope ($u_{ij}$). The data item is denoted by $i$ (e.g., restaurant), while $j$ denotes a data item value (e.g., Chinese restaurant). In a symbolic location model the data item value valid scope is represented by a set of logical IDs (e.g., the $cell_id$ of a cell in a cellular communication system), where the item value has valid answers. Since a data item may have different values in different cells, a data item is associated with a set of valid scopes, which is called the data item valid scope distribution $U_i$ (Zheng and Lee, 2002; Xu et al., 2003; Drakatos et al., 2006). A scope distribution may be shared by several data items. In a large-scale information system, the number of scope distributions can be very large. Every data server keeps a complete copy of the database, i.e., the same data items are replicated on all the data servers but probably with different values in different data servers. That is because every data server supports different service
areas hence, a LDD query will produce different answers from each data server.

**Example 4.1 (Valid scope).** For a data item \( (i) \), \( u_{i,A} \) denotes the valid scope of item value A and \( u_{i,B} \) denotes the valid scope of item value B. \( i = 1 \) denotes a restaurant data item (Example 3.2) with A to be a Chinese restaurant and B a Latin restaurant. If item value A is found in cells 1 and 2, \( u_{i,A} = \{1,2\} \). Likewise, if item value B is found in cells 3 and 4, \( u_{i,B} = \{3,4\} \). The scope distribution \( U_i \) of the restaurant database item \((i)\) with only two item values A and B is \( U_i = u_{i,A} + u_{i,B} = \{1,2,3,4\} \).

**4.2. Movement pattern – future cells for prefetching**

In order to form the context-aware cache with high benefit (value) entries, we first need to select a cluster of cells composed of \( k \) rings. For example if \( k = 3 \) the cluster consists of three rings for a total of \( 3k^2 + 3k + 1 = 37 \) cells (Fig. 2). Next, we can choose the Bordering Neighboring Group (BNC) of cells and the Non-Bordering Neighboring Group (NBNC) group of cells. The cells belonging to the BNC group are considered to be the most likely to be visited neighbors and thus have the highest probability of being the future location of the MT. Therefore, data items belonging to this range should be more likely to be marked for cache prefetching. The cells belonging to the NBNC group represent the less likely to be visited and thus, have a much smaller probability of being the next location and, therefore, have a much smaller prefetching benefit. However, cached segments associated with these NBNC cells are the best candidates for eviction when the cache has run out of space. Using the neighboring list of cells, we can provide a detailed calculation of the probability distribution for the next possible future cells and use this distribution to define the prefetching zone (PZ), with a given confidence level. It must be noted though that considerable processing power is needed to do the calculations resulting in a slow implementation and a considerable amount of memory space and bandwidth usage associated with prefetching information associated with all these cells. To mitigate this problem and also to improve the accuracy of the future location prediction, the proposed prefetching policy uses \( U_i \) as a masking operator applied on the neighboring list of cells denoted by \( C_i \) to derive a subset of cells called the prime list denoted by \( P_i \) where \( P_i = C_i \cap U_i \). Next, Example 4.2, uses a cluster of cells composed of three rings and an arbitrary scope distribution \( U_i \) to demonstrate how we derive the prime list of cells.

**Example 4.2 (Neighboring cells identification).** Depending on the mobile user’s current cell \( x \) and the movement direction selected, Table 2 shows the displacement number which is added to the user’s current cell number \( x \) to identify the next cell the MT moves to. The values of \( k = 1 \), 2 and 3 denote rings 1, 2 and 3 respectively. The displacement is inherited to the geometrical grid and is used to form a \( 15 \times 15 \) grid of cells (Fig. 4 depicts a \( 7 \times 7 = 49 \) cells portion of the total grid). The MT may move from the current cell (let us assume that \( x = 109 \)) to cells located at N, NE, SE, SW, NW. If the next selected direction is N, the next step will be to cell 94 (\( x + 15 = 109 + 15 = 94 \)). For the NE direction, the next cell will be 102 (\( x + 7 = 109 + 7 = 102 \)). For the SE direction, the next cell will be 117 (\( x + 8 = 109 + 8 = 117 \)). For the SW direction, the next cell will be 116 (\( x + 7 = 109 + 7 = 116 \)) and finally for the NW direction, the next cell will be 101 (\( x + 8 = 109 + 8 = 101 \)). Then a valid scope distribution \( U_i \) is assumed. This distribution is the sum of all item values valid scopes requested, i.e., the restaurant’s scope distribution to which the continuous query is referring to. The following equations show the calculations for the neighboring and prime lists of cells:

![Fig. 4. A four ring cluster (7 x 7 grid) of valid cells.](image-url)

| Table 2 |

| Neighboring valid cells identification |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| \( R_{k,1} \)   | \( x - 15 \)    | \( x - 8 \)    | \( x - 7 \)    | \( x + 7 \)    | \( x + 8 \)    | \( x + 15 \)    |
| \( R_{k,2} \)   | \( x - 30 \)    | \( x - 23 \)   | \( x - 22 \)   | \( x - 16 \)   | \( x - 14 \)   | \( x - 1 \)     |
| \( x + 1 \)     | \( x + 14 \)    | \( x + 16 \)   | \( x + 22 \)   | \( x + 23 \)   | \( x + 30 \)   |                |
| \( R_{k,3} \)   | \( x - 45 \)    | \( x - 38 \)   | \( x - 37 \)   | \( x - 31 \)   | \( x - 29 \)   | \( x - 24 \)    |
| \( x - 21 \)    | \( x - 9 \)     | \( x - 6 \)    | \( x + 6 \)    | \( x + 9 \)    | \( x + 21 \)   |                |
| \( x + 24 \)    | \( x + 29 \)    | \( x + 31 \)   | \( x + 37 \)   | \( x + 38 \)   | \( x + 45 \)   |                |

\footnote{The scope distributions of the data items are maintained and updated periodically by the database server.}
Algorithm 1 Candidate Queries List Formation

Sub-Algorithm

/* LA denotes the list of past requested attributes for the relation of \( Q_i \), ordered by descending frequency. LQ denotes the set of candidate for prefetching queries and LP denotes the list of past predicates for the relation of \( Q_i \), ordered by descending frequency. */

1: begin
2:  Case 1. Add one frequently requested attribute at a time
3:    for each attribute \( a \) in \( Q_i \) do
4:      \( Q_i = \text{add } a \) to requested attributes of \( Q \)
5:    end for
6:  if \( Q < > Q \)
7:    \( Q \) to LQ
8: end if
9:  Case 2. Add one frequently requested predicate at a time
10: for each predicate \( p \) in LP
11:    \( Q_i = \text{add } p \) to predicates of \( Q \)
12: end for
13: if \( Q < > Q \)
14:  \( Q \) to LQ
15: end if
16: Case 3. Remove one predicate of \( Q \) at a time
17: for each predicate \( p \) in \( Q \)
18:   \( Q_i = \text{remove } p \) in \( Q \)
19:   \( Q_i \) to LQ
20: end for
21: Case 4. Remove a predicate from \( Q \) and at the same time
22: add an attribute (combination of above cases)
23: for each predicate \( p \) in \( Q \)
24:  for each attribute \( a \) in LA
25:   \( Q_i = \text{remove } p \) in \( Q \)
26:   \( Q_i = \text{add } a \) to requested attributes of \( Q \)
27: end for
28: end for
29: \( Q \) to LQ
30: end

4.3. Query pattern and cache management

A context-aware prefetching action can be examined by its two components – (i) location context and (ii) query context. In the previous section, we defined the future cells of prefetching interest, while here we use the query context to granulate and filter the information inside these cells. In an LBS application, the service answer depends on the user’s context (e.g., time and location) from which the user issues a query. In a mobile scenario, this implies that the service must be able to refresh the answer to a query that is still active, when a change in the user context invalidates the previously provided answer. In the real world, as mobile users are changing locations, most likely they tend to ask about the same data item a few times until switching to another one. Affinity is a preference towards a particular result. Once the mobile client has demonstrated affinity towards one particular data item type (i.e., restaurant), each additional data item type has a lower affinity value for the duration of the query. Additionally, a certain user may query the same item often, while another user may query the same item occasionally. Thus, both a user’s movement pattern and the query pattern need to be considered to further improve the effectiveness of a cache prefetching policy. Assume that a new query \( Q \), “give me all the names of the Chinese restaurants at the current location in the medium price range” is being processed where a prefetching decision needs to be made. In trying to design the optimum query for prefetching we identify two main issues: (i) how to form a list of candidate queries for prefetching and (ii) examine the candidates list to identify the best candidates for prefetching.

First Issue: Form a candidate queries list for prefetching based on the current query \( Q \) semantic description. We do so by examining the query history to find the most frequently requested attributes and query predicates, and then try to augment or alter the current query using this history. To limit the space of candidate queries we only consider the following cases which are intuitive and offer good bandwidth and cache space savings: (i) augment \( Q \) by a single requested attribute (e.g., ask for working times in addition to prices), (ii) add a predicate (e.g., price="medium" in addition to the location predicate), (iii) remove a predicate (e.g., get all local restaurants instead of only the Chinese), and (iv) remove a predicate and add a requested attribute (combination of (i) and (iii)). Based on this described logic, the sub-algorithm (Algorithm 1) is designed to form a list of candidate queries LQ for prefetching. Table 3 lists part of the algorithm’s output using query \( Q \) for input. Additionally, Table 4 lists the attributes and predicates of the candidate queries list.

Second Issue: Examine the candidate queries list for the best query choice for prefetching. In predicting the user’s next query, we can make the following observations regarding the prefetching cost of the candidate queries list. Considering prefetching to be a form of caching for dynamically generated content (Datta et al., 2003), one can also consider prefetching along a number of granularity levels such as the cell, relation, attribute and item values (Cell → Data_item → Attribute → Item_Value). In predicting the user’s next query, we can make the following
observations regarding the prefetching cost of the above listed candidate queries and use two criteria to determine the benefit of prefetching a query.

\[(\phi_1) \text{ The attributes commonality criterion is satisfied if the candidate to prefetch query } Q_i \text{, shares attributes with the current query } Q.\]

\[(\phi_2) \text{ The tuple similarity criterion is satisfied if the result tuples of } Q_i \text{, are a subset of the result tuples of } Q.\]

The best candidates for prefetching are the queries \(Q_i\) that satisfy both criteria \((\phi_1, \phi_2)\). We assign a high affinity level to this query, \(f = \text{High}\). If we still have cache space available, we prefetch queries that satisfy either one of the two criteria \((\phi_1, \phi_2)\) (medium affinity level, \(f = \text{Medium}\)). Finally, we may also prefetch queries that partially satisfy the two criteria (low affinity level \(f = \text{Low}\)).

In trying to anticipate the future user needs a replacement policy must also be defined to make room for new data items when the cache becomes full. In previous work (Drakatos et al., 2006) we described a compatible future location-aware replacement policy that is used here with the prefetching strategy when the cache is full. Table 5 lists the candidate queries using a bandwidth and space allocation cost taxonomy. The overall steps for the proposed prefetching strategy are shown in Algorithm 2. Using the above described criteria, we note that the first query \(Q_1\) in the candidate list is based on the same location and data item type (relation) and it meets both criteria. Therefore, it only marks some of the resulting tuples of \(Q\). It clearly represents the cheapest choice as far as network resources are concerned and it will achieve a good prediction level for the next query. As a second best choice for prefetching query selection, queries \(Q_2\) and \(Q_3\) meet only one of the two criteria. \(Q_2\) asks for a new attribute and \(Q_3\) brings new tuples. As a last choice notice that \(Q_7\) asks for a new attribute and will also bring new tuples. Regarding queries with a “NO” in the savings column, queries \(Q_9, Q_4\) and \(Q_5\) are new queries that ask for new attributes and that will have to retrieve new tuples at another location; therefore, there is going to be a higher increase in saving cost and bandwidth. The case of a join query type may be treated as multiple queries, each against a different relation.

Algorithm 2 Prefetching overall algorithm

/* \(Q\) denotes the candidate queries for prefetching\n\(Q\) denotes the current query
\(U_i\) denotes the data item valid scope
\(\phi_1\) – The attributes commonality criterion indicates \(Q_i\) has subset of attributes of \(Q\) \((\forall i \subseteq \forall a)\)
\(\phi_2\) – The tuple similarity criterion indicates \(Q_i\) has subset of tuples of \(Q\) \((\forall \text{Ans}(Q) \subseteq \text{Ans}(Q))\)*/

Step 1. Form the Prime List of Cells (Prefetching Zone)

\[P(prime) = C(candiate) \land U_i\]

Step 2. Form a List of Candidate Queries

Use Sub-Algorithm 1 to form the list of candidate queries \(LQ\) for prefetching.

Step 3. Select the Optimum Query Context

Use the two criteria \((\phi_1 \text{ and } \phi_2)\) to compare \(Q_i\) with \(Q\)

While cache space is available

Select queries \(Q_i\) that satisfy both criteria

Select queries \(Q_i\) that satisfy one of the two criteria

Select queries \(Q_i\) that partially satisfy the two criteria

End While

Step 4. Augment Optimum Query and Prefetch

Use the PZ future cells to augment selected \(Q_i\rightarrow Q_i\)

Prefetch using augmented queries \(C_{prime}\)

End

Table 3

<table>
<thead>
<tr>
<th>Query candidates list for prefetching</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Q): “give me all the names of the Chinese restaurants at the current location in the medium price range”</td>
</tr>
<tr>
<td>(\bigbullet Q1): “give me all the names of the Chinese restaurants at the current location that open from 6:00 p.m. to 10:00 p.m. in the medium price range”</td>
</tr>
<tr>
<td>(\bigbullet Q2): “give me all the names and price ranges of the Chinese restaurants at the current location in the medium price range”</td>
</tr>
<tr>
<td>(\bigbullet Q3): “give me all the names and price ranges of the Chinese restaurants at the next location”</td>
</tr>
<tr>
<td>(\bigbullet Q4): “give me all the names of the Chinese restaurants at the current location”</td>
</tr>
<tr>
<td>(\bigbullet Q5): “give me all the names of the Latin restaurants at the current location”</td>
</tr>
<tr>
<td>(\bigbullet Q6): “give me all the names and price ranges of the Chinese restaurants at the current location”</td>
</tr>
<tr>
<td>(\bigbullet Q7): “give me all the names and price ranges of the Chinese restaurants at the current location”</td>
</tr>
</tbody>
</table>

Table 4

<table>
<thead>
<tr>
<th>Candidate queries-attributes and predicates description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Q)</td>
</tr>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
5. Prefetching cost analysis model

In a mobile wireless environment, the most scarce and critical resources are those strictly connected to the use of portable devices to access the information service, such as disk space, processing power, wireless link bandwidth and amount of energy. In this paper, we focus our attention on the bandwidth and define it as the cost of prefetching the consumption of this scarce resource. As a measure of bandwidth occupancy, we consider the number of bytes per second (W) that traverse in both directions in the wireless link, while a query is active. In a mobile environment, the user is unaware from where the information is received; however, two events may take place; a cache hit, and a cache miss. For the mobile environment, an out-of-scope condition is the time the MT has moved into a new cell outside the prefetched cluster, while an information request is still “active”. Pmiss is the probability of not finding locally the required information when an out-of-scope condition occurs (cache miss), while the probability of a cache hit is denoted by 1 – Pmiss.

During a cache miss, the information is retrieved from a remote source in trem time units (e.g., a server or a proxy server), and during a cache hit the information is retrieved locally in tlc time units. The latency, or response time, is the time from the submission of the query to the time when the result is obtained. The latency is often associated with a high variability in transfer times for the same data item and makes cache management decisions more difficult. Other contributions to the latency, such as the time needed to detect the out-of-scope condition, are not considered. Instead, the focus is on the time needed to retrieve and start loading the information from the data server responsible for the particular information service trem. This time depends on the server load, the amount of information to be transferred and the available bandwidth. Since the bandwidth of a wireless channel is usually quite low, the value of trem can be excessively high. Prefetching is considered when a cache miss occurs. Clearly, if we were concerned only with reducing the overall performance, all we would need to do would be to increase the frequency of prefetching and the amount of prefetched data. A compromise is needed between two opposite goals – (i) to keep Pmiss small and (ii) to optimize resource usage i.e. To determine an ideal number of cells to be used for prefetching. Therefore, we need to define a cost evaluation model and try to find the equilibrium between cost and performance. The average occupation of the wireless link can be expressed as W = IPC + TNM. The IPC = Wq + WaN0 component, denotes the initial prefetching cost and TNM = tqbσ(Wq + N, Wq) denotes the cost of the total number of misses. We may, thus, write the expression for W as follows:

\[ W = (W_q + W_a N_0) + t_q b_k \sigma_k (W_q + W_a N_k), \]

where Nk is the number of cells for prefetching when the ith cache miss occurs. For a standard random movement Nk = 3k^2 + 3k + 1. N0 is the number of cells used for prefetching when the user submits the query the first time. The term t_q b_k \sigma_k = t_q P_{miss} represents the number of missed hits during the active period of the query Q denoted by t_q. The transition from state k to state 0 represented by b_k = \frac{2(k-1)!}{k!}, using the state diagram shown in Appendix B, corresponds to the exit from the cluster covered by previously prefetched information and to the reconstruction of a new cluster.

Next, we need to evaluate both the transition and steady state probabilities. We use a discrete time Markov model that describes the motion of the user within the area “covered” by the prefetched information (i.e., within cluster of k rings) starting from the central position, which is the mobile user’s current cell. The state diagram derivation of the state transition probabilities is given in Akyildiz et al. (1996), Levine et al. (1997) and Persone et al. (1998), where state i means that the user is in a cell belonging to ring i (0 ≤ i ≤ k) and a_i, b_i and c_i are the probabilities that, at the end of each time slot, the user moves to ring i – 1, i + 1 or remains in ring i, respectively. The user moves to an adjacent cell with probability γ, or remains in the same cell with probability 1 – γ. Given the uniform motion model, these values can be calculated by counting how many edges are adjacent to each polygon that belongs to ring i. σ_k denotes the steady state probability of being in ring i when a cluster of radius k is reconstructed each time a local storage miss occurs. Previous research using the Markov model (Levine et al., 1997; Persone et al., 1998) has shown an approximate expression for σ_k that can be calculated in the case of the two-dimensional motion as follows:

\[ \sigma_k = \frac{(\frac{b^2}{b^2 - a^2})^2 (\frac{b^2}{b^2 - a^2})^k}{(b^2 - a^2 - ab + ybk - yak) + (a^2 - ab + a^2)}. \]
Finally, we derive Eq. (2) making the assumption that the number of cells used for prefetching when the user submits the query the first time is equal to the number of closest cells used for prefetching thereafter

\[ W = (W_d + W_m N_k)(1 + t_d \beta_d). \]  

(2)

It should be noted, that the above formulation represents only one possible way to formulate the actual data traffic. For example, it does not consider explicitly the number of bytes exchanged to set up the connection between the user device and the data server. This number depends on the particular transport protocol used for the connection, and could be included in the model; however, it will require more complex formulations.

5.1. Numerical results

In this subsection, we present numerical results obtained by using the following values: \( t_{\text{req}} = 10 \text{ ms} \) and \( t_{\text{rem}} = 5 \text{ s} \). We consider an average query duration \( t_q = 500 \) discrete time units and \( W_d = 10 \text{ Kb} \) as the typical query results. From Figs. 5 and 6, we note that using clusters with radius \( k = 1 \) (6 cells) for prefetching we experience a latency (time delay) improvement with no significant performance cost increase. However, when using clusters with a radius \( k > 1 \) (more than 6 cells), there is a wireless bandwidth increase for slow motion (mobility factor \( \gamma = 0.2 \)), which is even more significant for fast movement (mobility factor \( \gamma = 0.8 \)). Using the uniform random movement model a normal prefetching strategy would prefetch information associated to a cluster of rings \( \{0, 1, 2, 3, \ldots, k\} \) centered at the MT’s current cell for a total number of cells given by: \( N(k) = 3k^2 + 3k + 1 \). These results suggest that anticipating prefetching to prevent a cache-miss occurrence leads to a significant performance advantage, but with an increase of wireless bandwidth consumption, especially for \( k > 1 \).

The proposed method, while maintaining the prefetching cost savings large enough for better performance, at the same time uses a modified random movement model to mitigate the prefetching cost performance issue. The results of this set of experiments demonstrate that the proposed modified random movement model in both cases (slow and fast motion) is able to significantly reduce the prefetching cost. This happens because, for the same scope of prefetching (i.e. same number of rings, \( k \)), our model’s valid scope factor will use a reduced number of cells (\( P_i(k) \)). Remembering that \( P_i(k) \) is the prime list (Section 4.2) of cells which is a subset of \( N(k) \). In a large-scale information system, the number and size of scope distributions can be very large. Assuming that for a certain data item the scope distribution size expressed in the number of cells covers 75% of the neighboring cells (e.g. the number of restaurants in a metropolitan location). To quantify this prefetching cost savings let \( \epsilon \) be the percentage of cells reduction index, which is specific to the data item scope distribution, in this case \( \epsilon = 25\% \). Then, we can estimate using Eq. (2) the prefetching cost savings. Fig. 6 shows that for \( \epsilon = 25\% \) the achieved decrease in prefetching cost expressed versus the number of rings of cells used for prefetching, varies from 0% to 33% for the slow motion case and from 0% to 40% for the high mobility case.

6. Experimental study

6.1. The environment

In our simulation, we model a wireless infrastructure based network. The simulated area in which mobile user’s movement takes places, is described using a \( N \times N \) rectangular area of cells (Fig. 7), represented by fixed size hexagons (other shapes may also be used). The same cell array was adopted in previous works (Ho and Akyildiz, 1995; Akyildiz et al., 1996; Hac and Zhou, 1997). To set
the simulation environment, we investigated both using the network simulator (ns-2) (The Network Simulator, 2002) and developing a custom simulation environment. After analyzing the ns-2 environment, we found that to exploit ns-2 in our project, an enormous amount of changes should be applied to its internal classes. We, therefore, decided to implement our own simulation environment using C++. The simulation environment includes a heuristic mathematical model to identify the neighboring cells and efficiently measure the distance between any two cells. Cells are identified using both an absolute number, which is the cell number, and a reference number. The expressions for cell numbers and distance $d$ are easily derived from the proposed grid geometry and are shown in Appendix A. The distance is recorded in terms of the number of cells travelled by a mobile user at each trial. The simulation is carried out for the RW and DIR mobility models. The combination of these two models can be used to simulate realistic models such as daily, weekly and monthly in combination with user profiling. Under the RW model the MT selects a random cell as its destination and follows a random path to reach it. At each cell it pauses for a random period of time and may also select a different speed. The next step is always selected randomly. DIR restricts the selection of the next destination so that moving direction is roughly reserved. This characteristic if combined with the mobile user’s profile obviously is a better model for purpose movements.

6.2. The client cache model

Besides the random walk process described in the previous section, the mobile client is modelled with three independent processes: cache manager, query process and query generator. The query process continuously generates location-dependent queries for a number of data items types (up to 10) selected randomly. A query delay is implemented between each consecutive query. For our experiments we use queries which place simple equality conditions on the underlying data based on the user’s current cell. The client is assumed to have a cache of fixed size (Cache Size), which is a percentage of the database size. Two cache size settings are used; one from 1% to 10% and another one from 10% to 50% of the database size. To answer a query, the client first checks its local cache. If the data value for the requested item with respect to

Fig. 7. A $N \times N = 15 \times 15$ grid of cells used by the mobility model.
the current location is available, the query is satisfied locally. Otherwise, additional data is fetched from the server. The Query Generator generates queries according to the scope distribution used by the different experiments. It also changes the query delay patterns. Several elements in the simulation need to be generated randomly with each request.

The affinity effect is created by the query generator using a calculated query pool. The query pool defines how large the random number needs to be. Creating larger and larger pools of numbers that will result in selecting a particular queryType. Next, we get a random number in the range 1 to queryPool inclusively. To simulate the affinity towards a selected query properly, we have used the Matsumoto and Nishimura’s random number generator ‘Mersenne Twister’ (Matsumoto and Nishimura, 1998), which generates numbers in a much larger range. The movement process is controlled by a number of modules which uses a random process to select the next direction (cell). Other modules define the valid cells for each random walk. Instead of having one single cache a number of semantic caches are simulated by the cache management process, which also removes and adds segments based on the underlined cache management scheme. This approach implies some data item type differentiation, which is proven beneficial for processing data items of different types.

7. Performance evaluation

In this section, the proposed location and context aware cache management strategy (CA) is evaluated and compared with the standard directional scheme based on tangent velocity (Park et al., 2004) (denoted as non-CA or DR). The cache hit ratio \(h\) is used as the primary performance metric, because most of the other performance results can be derived from the cache hit ratio. The cache hit ratio is the percentage of all the requests that can be satisfied by searching the cache for a copy of the requested data item. Mobile client mobility is patterned using the random walk model (Section 3.1). Each simulation trial uses a set of valid cells and lasts until the MT has completed a fixed number of movements. The expected average hit ratio is estimated for each prefetching strategy of \(N\) iterations. For each run of the simulation different movement paths may be selected. For each iteration, the values for the hit ratios are added to the previous ones and at the end of the \(n\)th iteration (trial), we average the results for the proposed strategy called CA and DR. The DR scheme uses the FAR (Ren and Dunham, 2000) (currently used as the formal semantic caching replacement policy), when additional cache space is needed. The results are obtained when the system has reached a stable state. For each simulation trial a number of warm-up queries are issued, so that the warm-up effect on the client cache is eliminated before collecting the performance metrics. In a mobile environment, prefetching is considered when a cache miss occurs. Fig. 8 depicts an overall data caching system model used for this simulation. Since it is well-known that prefetching will benefit from large cache size we use small cache size from 1% to 10%. However, steadily falling costs of storage lead to caches of larger sizes (Podlipnig and Böszörményi, 2003), so we have included some experiments using larger cache size levels i.e., from 10% to 50%. In the next sub-sections, we investigate the performance of the examined schemes focusing on the characteristics that we ‘feel’ could impact the prefetching decisions, such as cache size, simulation time, the query delay, the data item types, the affinity factor, cells numbers and the movement speed.

Finally we note that our prefetching strategy fails for uncorrelated query patterns as expected. In such extreme cases prefetching is negative effect on the overall system performance. As it is noted in the cache prefetching components (Section 4), this paper focuses on the query and movement patterns. Furthermore, we believe that the third factor, i.e., the user profile can be included (to be examined in future work) as a means to mitigate this issue by providing a control facility for purpose movement (i.e., weekly monthly or daily schedules of operation).

7.1. Impact of cache size

In this set of experiments, we measure the cache hit ratio under five cache size settings: 2%, 4%, 6%, 8% and 10% of

---

**Fig. 8.** Simulation work flow (res: resident query, rq: remote query).
the database size. The mobility model is RW. The duration of the experiments is 5000 movements per trial. As expected, the cache hit ratio increases with increasing cache size for all schemes, as more important data items having high access probability (cache benefit) are stored in the cache. CA demonstrates (Fig. 9) an even greater performance advantage, approximately 0–20% compared to the standard direction policy (DR). Regarding the underlying replacement policy used by the cache, LRU is included here as a baseline scheme. The results for this scheme are compatible with the ones other researchers have demonstrated (i.e., previous work) from 5% to 15%. We believe that the performance advantage of CA is a result of CA being able to adapt itself to the data items distribution in the neighboring cells, due mainly to its data item scope distribution factor.

Fig. 10 demonstrates that LRU outperforms DR and CA under the directional mobility model (DIR). This is because under the DIR, a data item no longer accessed can be detected and removed from the cache more quickly than by LRU. For DIR (using FAR for replacement), the data item depends on its distance from the current location of the user and movement direction. This result is exactly the opposite for the RW model, where LRU might erroneously evict the items that are to be requested by the near future queries. Both CA and DR are independent of the recent access history, and therefore perform much better in this case. In summary, CA performs more stable than the other two schemes because it uses a larger scope (i.e. rings of cells) rather than continuous tangent velocity calculations, which are proven to be problematic for active mobile users.

Additionally (Fig. 10) demonstrates that the cache-hit-ratio for the RW model is larger than for the DIR model in all cache schemes. This is expected as RW exhibits better query locality than DIR; so there is a caching benefit in this scenario.

7.2. Impact of affinity factor

In this set of experiments, we simulate the effect of the query pattern by changing the affinity factor. The affinity factor (i.e., from $f = \text{Low}$, $f = \text{Medium}$ and $f = \text{High}$) determines how close the prefetching query matches the two criteria ($\phi_1, \phi_2$) defined in Section 4.3. While wandering, a mobile user may have an affinity (preference) towards a certain item type (continuous type of query). It is possible that the user could get sidetracked every now and then – those would be the times when they query for something else. Fig. 11 demonstrates that the higher the affinity value the better the performance results. This is due to the fact there is a much better future query pattern prediction and, therefore, the cache management strategy does a much better job in evaluating the data items cache value. The results of this experiment demonstrate that CA can reach very high cache hit ratios as the cache size increases. This characteristic suggests the possible capabilities of the proposed strategy in a larger design/application space, where more MTs can be used to see if they interface with one another or help each other in a peer-to-peer coopera-
tive computing architecture, i.e., if there is no data access via a BS or a MSS ask your peer.

7.3. Impact of data item types

In this set of experiments, the scalability of the proposed cache management strategy is evaluated by varying the data item types from one to five and comparing them to DR. Figs. 12 and 13 show the performance results when we change the number of data item types from “1” to “5”. The performance of CA shows on average a 10–20% better performance compared to DR, depending on the number of different types of data items used in the database, however, it decreases significantly as the number of data items increases. It is interesting to notice that the DR method is hardly affected by the number of data item types. This happens because CA explores its future location capabilities in a 360° global scope, while DR uses an implied future prediction mechanism (tangent velocity calculations) only to the cells in the exact opposite direction of the movement. This scheme has limited success and does not improvise for sharp turns (really random movements) of the MT. In addition, DR demonstrates a significantly higher overhead of continuous velocity calculations every $\Delta t$. However, CA requires an insignificant overhead to identify the mobile user’s neighboring cells and to estimate the replacement value of the cached segments. From these results we can even draw the conclusion that the proposed method is more appropriate for a LBS system where the database is partitioned to different volumes per data item types for higher performance.

7.4. Impact of query delay

The query delay is the time interval between two consecutive client queries. We do not necessarily have to query for any item each time a new movement is chosen. In this set of experiments we evaluate the impact of the query delay on the cache hit ratio under two contending prefetching schemes. A query delay “1”, means that the query takes place with every random movement, a query delay “2”, means that the query takes place with every other random movement, etc. As illustrated, when the query delay is increased from “1” to “5”, the standard DR scheme demonstrates an approximate 18.5% cache hit ratio decrease. This is expected because, for a longer query interval the client would make more movements between two successive queries; thus the client has a lower probability of residing in one of the valid scopes of the previously queried data items when a new query is issued. However, interestingly enough the proposed strategy responds differently by displaying almost no change in the cache hit ratio (only a 2% decrease) as the query delay increases from “1” to “5”.

DR (Fig. 15) blindly prefetches information content based on the user’s direction that needs to keep evaluating
every $\Delta t$ time interval, hence it cannot prefetch far enough. However, CA (Fig. 14) can do more selective prefetching which allows it to prefetch at longest distance. To further validate these results we run the experiment for the regular database size (CA-1, DR-1) and then for a double database size (CA-2, DR-2) with approximately the same results (Fig. 16). This experiment allowed us to quantify the significant advantage of CA in a real life random query delay scenario over the standard DR scheme.

### 7.5. Impact of movement speed

The movement velocity is simulated using a maxspeed defined by the total number of movement for the duration of the simulation, i.e. maxspeed = 5000 time steps, approximately 900 s. The cache hit ratio results are plotted against a scale from 10% to 100% of the maxspeed. Fig. 17 shows cache average response time against the cache sizes 10%, 30% and 50% of the database size for the CA and DIR schemes. Fig. 18 shows cache average response time against a number of speed values (i.e., 25%, 50% and 75% of the maxspeed) for all three competing policies DR, CA and LRU. As it is expected as the speed of movement increases both DR and LRU experience a substantial performance reduction, which of course increases as the cache size increases. CA demonstrates a much less dramatic performance reduction for higher speed which we consider a significant advantage over the other policies, whereas DR demonstrates a better performance at lower speed of movement as it was noted before this scheme does not perform well in a fast moving environment.

### 8. Conclusions and future work

In a mobile computing paradigm, caching alone is generally not enough to improve the performance of mobile
systems. Moreover, prefetching has a broader application range than simply storing already used data in the cache. In this paper, we have considered both the movement pattern and the query pattern. Noticing that the query has valid answers only within a set of cells predefined for each data item (item valid scope), we first proposed an efficient future cell prediction filtering mechanism based on the valid scope concept. Next, we improved the prediction level into the most likely future query pattern based on the query affinity, while we preserved the cache content with the highest benefit. Both the performance and simulation models are based on general “context” and “user behavior” models. From the above performance results we can draw the conclusion that for a context-aware information service, prefetching is highly beneficial for both latency and traffic reduction. However, in some cases prefetching is only beneficial for latency reduction because it causes a cost (bandwidth cost) increase with respect to no prefetching. We have presented a context-aware future location prediction approach to mitigate this issue based on the data items valid scope and query content control. Regarding the future direction of this research, we plan for a more rigorous query pattern prediction by incorporating the user profile and a more realistic mobility model.

Acknowledgements

The authors would like to acknowledge Carol Choi and Phong Luu for their programming support in the simulation part of this work under NSF Grant Nos. 212400512 and 212400513 and Dr. Zuji Mao (Lucent technologies) for his help with the formulation of the main concepts of this research. Also, the authors are very grateful to the anonymous reviewers for their comments and critique which improved the quality of this paper significantly.

Appendix A. Cell numbers and the distance formula

The Cell number: Using a $15 \times 15 = 225$ grid, cell numbers are assigned for each cell. For each cell number “$C$”, the row and column are given by the following formula:

$$
\text{Row}(C) = \text{int}((C - 1)/15) + 1,
$$

(A.1)

$$
\text{Col}(C) = \begin{cases} 
X \times 2, & X < 8, \\
(X - 8) \times 2 + 1, & \text{otherwise},
\end{cases}
$$

(A.2)

where $X = C \mod 15$.

The relative distance number: In addition to the cell number the relative distance number is inherent to the proposed grid structure for each cell number. Relative distance numbers are used in the simulation to identify the neighboring cells based on the mobile user’s current cell.

The distance $d$ between two cells: The expression for distance $d$ is easily derived from the proposed grid geometry. This expression is used to determine the location distribution of a mobile user. The distance is recorded in terms of the number of cells travelled by a mobile user at each trial. Assuming that a cell $(x_1,y_1)$ is $d$ cells away from another cell $(x_2,y_2)$, the $d$ is given by

$$
d = \begin{cases} 
|x_1 - x_2|, & \text{if } |y_1 - y_2| = 0, \\
|y_1 - y_2|, & \text{if } |x_1 - x_2| = 0, \\
|x_1 - x_2| + |y_1 - y_2| - D, & \text{Otherwise},
\end{cases}
$$

(A.3)

where $D$ denotes the ring distance between the two cells defined by

$$
D = \frac{1}{2} \min(|x_1 - x_2|,|y_1 - y_2|).
$$

Example (Cell numbers). Using Eqs. (A.1) and (A.2) the row and column numbers for two random cell numbers 109 and 125, are $(8,8)$ and $(10,9)$ respectively. Next using Eq. (A.3), the shortest path between the two cells is given by $(|8 - 10| + |8 - 9|) - 1 = 2$ cells. Also notice that the relative distance numbers are 0 and +16, respectively.

Appendix B. Transition probabilities evaluation

A MT moves to a neighboring ring cell with probability $1/6$. In Fig. B.1, for the MT in a cell marked (*) at ring $i$, where $(1 \leq i \leq d)$, the MT moves to a layer $i - 1$ cell with probability $1/6$ (the cell has one bordering line for the $(i - 1)$st neighbor, or moves to a ring $i$ neighbor with probability $1/6 + 1/6 = 1/3$ (the cell has two bordering lines with $i$ neighbors), and finally moves to a neighboring higher ring $i + 1$ with a probability $1/6 + 1/6 + 1/6 = 1/2$ (the cell has three bordering lines for $i + 1$ neighbors).

If the MT is a ring cell without the * (see Fig. B.1) mark at ring $i$, then it moves to ring $(i - 1)$, $i$, or $(i + 1)$ with the same probability $1/6 + 1/6 + 1/6 = 1/3$. Thus, we will have the following three movement scenarios for the MT located at the ring $i$:

1. The MT moves to ring $i - 1$ with a probability

$$
\left(\frac{6}{6i}\right) \left(\frac{1}{6}\right) + \left(\frac{1}{6} - \frac{6}{6i}\right) \left(\frac{1}{3}\right) = \frac{2i - 1}{6i} = \frac{1}{3} - \frac{1}{6i},
$$

Fig. B.1. Layers crossing probabilities.
2. The MT moves to a higher layer \((i + 1)\) with probability 
\[
\left( \frac{6}{6i} \right) \left( \frac{1}{2} \right) + \left( \frac{1 - \frac{1}{6i}}{6} \right) \left( \frac{1}{3} \right) = \frac{2i + 1}{6i} = \frac{1}{3} + \frac{1}{6i}.
\]
3. The MT moves to a cell within the same ring \(i\) with probability \(1/3\).

At the end of each timeslot, the user moves to an adjacent cell with probability \(\gamma\), or remains in the same cell with probability \(1 - \gamma\). Then \(a_i, b_i\) and \(c_i\) are the probabilities that, at the end of each timeslot, the MT moves to ring \(i - 1, i + 1\) or remains in ring \(i\), respectively. These probabilities are given below:

\[
a_i = \begin{cases} 
0 & \text{if } i = 0, \\
\left( \frac{i - 1}{6i} \right) & \text{if } i > 0,
\end{cases} \tag{B.1}
\]

\[
b_i = \begin{cases} 
\gamma & \text{if } i = 0, \\
\left( \frac{i + 1}{6i} \right) & \text{if } i > 0,
\end{cases} \tag{B.2}
\]

\[
c_i = \begin{cases} 
1 - \gamma & \text{if } i = 0, \\
\left( 1 - \gamma \right) + \frac{1}{3} & \text{if } i > 0.
\end{cases} \tag{B.3}
\]

The state diagram derivation of the state transition probabilities given in Akyildiz et al. (1996), Levine et al. (1997) and Persone et al. (1998) is shown in Fig. B.2, where state \(i\) means that the user is in a cell belonging to ring \(i\), \(0 \leq i \leq k\).

References


Stylianos Drakatos has more than 15 years of experience (Including Big Five) in the field of Telecommunications and Information Technology. With Bell South, he participated in the design and implementation of a number of technology integration projects, as a member of the planning and re-engineering team. He also has worked with several other important companies such as Price Waterhouse in New York City as Principal Consultant and Florida Power & Light Co. in Miami, Florida. He is
Niki Pissinou received her Ph.D. in Computer Science from the University of Southern California, her M.Sc. in Computer Science from the University of California at Riverside, and her B.S.I.S.E. in industrial and Systems Engineering from the Ohio State University. She is currently a tenured professor and the director of the Telecommunications & Information Technology Institute at the Florida International University in Miami, Florida. Previously, she was tenured faculty at the Center for Advanced Computer Studies at the University of Louisiana at Lafayette where she was also the director of the Telecommunication & Information Technology Laboratory partially funded by NASA, and co-director of the NOMAD: A wireless and Nomadic Laboratory partially funded by NSF, and the advanced Network Laboratory. Dr. Pissinou is active in the fields of computer networks, information technology and distributed systems. Dr. Pissinou has published more than 100 refereed publications and has received best paper awards. She has also co-edited eight volumes and is the author of an upcoming book on wireless Internet computing. She has received extensive funding for her research work from agencies such as NSF, NASA, DAPRA and ARO including two recent NSF awards on Wireless Networks.

Kia Makki received his Ph.D. in computer science from the University of California, Davis, and two M.S. degrees from The Ohio State Universities and West Coast University. He is currently a tenured professor with an endowed chair professorship and an Associate Dean of the College of Engineering at the Florida International University. Previously he was tenured faculty with an O’Krepki Chair professorship at the University of Louisiana, Lafayette, the Director of the Center For Telecommunications Studies, and the Co-Director of the NOMAD and Advanced Computer Laboratory which is partially funded by NSF and NASA. His technical interests include wireless and optical telecommunications. He has numerous refereed publications and his work has been extensively cited in books and papers. Dr. Makki has been Associate Editor, Editorial Board Member, Guest Editor of several International Journals including Editorial Board Member of several telecommunication journals. Dr. Makki has been involved in numerous conferences as a Steering Committee Member, General Chair, Program Chair, Program Vice-Chair and Program Committee Member.

Christos Douligeris received the diploma in electrical engineering from the National Technical University of Athens in 1984 and the M.T., M.Phil., and Ph.D. degrees from Columbia University in 1985, 1987, and 1990, respectively. He has held positions with the Department of Electrical and Computer Engineering at the University of Miami, where he reached the rank of associate professor and was the associate director for engineering at the Ocean Pollution Research Center. He is currently an associate professor in the Department of Informatics at the University of Piraeus, Greece. He has served on technical program committees of several conferences. His main technical interests lie in the areas of performance evaluation of high speed networks, security in networking, resource allocation in wireless networks, information management, risk assessment, and evaluation for emergency response operations. He is an editor of IEEE Communications Letters, Computer Networks, IEEE Network, a former technical editor of IEEE Communications; a senior member of the IEEE; and a member of the ACM and the Technical Chamber of Greece.