11-3-2006

Context-aware data caching for mobile computing environments

Stylianos Drakatos
Florida International University

DOI: 10.25148/etd.FI15101224

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

CONTEXT-AWARE DATA CACHING FOR MOBILE COMPUTING ENVIRONMENTS

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Stylianos Drakatos

2006

To: Dean Vish Prasad
College of Engineering and Computing

This dissertation, written by Stylianos Drakatos, and entitled Context-Aware Data Caching for Mobile Computing Environments, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Kia Makki, Co-Major Professor

Kang Yen

Christos Koulamas

Christos Douligeris

Niki Pissinou, Co-Major Professor

Date of Defense: November 3, 2006

The dissertation of Stylianos Drakatos is approved.

Dean Vish Prasad
College of Engineering and Computing

Dean George Walker
University Graduate School

Florida International University, 2006

ACKNOWLEDGMENTS

I would like to express my deep thanks and appreciation to my co-advisors professor Dr. Niki Pissinou and professor Dr. Kia Makki for their dedicated guidance, advisement, and support; I acknowledge that what I learned from them will benefit my advancement in the rest of my life. I am deeply indebted to them for their professional guidance and encouragement. My meetings with them have been extremely rewarding because of their insight, knowledge and quick grasp of essentials of my research area. Their excellent criticisms, and their prompt, careful reading of early drafts of this thesis were enormously helpful in its clarification and development.

I am grateful to Dr. Christos Douligeris for his continuous support, technical ideas and multiple paper reviews. My sincere thanks are also due to the other two members of the committee, Dr. Kang Yen, and Dr. Christos Koulamas, for their valuable comments and suggestions. Finally I would like to thank my parents for their ever continuing guidance and encouragement, especially my brother Dr. Charalambos Drakatos for never doubting the results of my efforts.

ABSTRACT OF THE DISSERTATION

CONTEXT-AWARE DATA CACHING

FOR MOBILE COMPUTING ENVIRONMENTS

by

Stylianos Drakatos

Florida International University, 2006

Miami, Florida

Professor Niki Pissinou, Co-Major Professor

Professor Kia Makki, Co-Major Professor

The deployment of wireless communications coupled with the popularity of portable devices has led to significant research in the area of mobile data caching. Prior research has focused on the development of solutions that allow applications to run in wireless environments using proxy based techniques. Most of these approaches are semantic based and do not provide adequate support for representing the context of a user (i.e., the interpreted human intention.). Although the context may be treated implicitly it is still crucial to data management. In order to address this challenge this dissertation focuses on two characteristics: how to predict (i) the future location of the user and (ii) locations of the fetched data where the queried data item has valid answers. Using this approach, more complete information about the dynamics of an application environment is maintained.

The contribution of this dissertation is a novel data caching mechanism for pervasive computing environments that can adapt dynamically to a mobile user's context. In this dissertation, we design and develop a conceptual model and context aware protocols for wireless data caching management. Our replacement policy uses the validity of the data fetched from the server and the neighboring locations to decide which of the cache entries is less likely to be needed in the future, and therefore a good candidate for eviction when cache space is needed. The context aware driven prefetching algorithm exploits the query context to effectively guide the prefetching process. The query context is defined using a mobile user's movement pattern and requested information context. Numerical results and simulations show that the proposed prefetching and replacement policies significantly outperform conventional ones.

Anticipated applications of these solutions include biomedical engineering, telehealth, medical information systems and business.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

LIST OF ACRONYMS

ATM: Asynchronous Transfer Mode

BS: Base Station

CS: Caching Strategy

CAC: Context-Aware Cache

CA: Context-Aware

DBMS: Data Base Management System

DOMINO: Data Base for Moving Objects

LDD: Location Dependent Data

FLA: Future Location-Aware

FAR: Furthest Aware Replacement

GSM: Global System for Mobile Communications

GPS: Global Positioning Systems

LRU: Least Recently Used

LA: Local Area

LBS: Location Based Services

LDIS: Location Dependent Information Services

MOST: Moving Objects Spatio-Temporal

MRU: Most Recently Used

MSC: Mobile Switching Center

MC: Mobile Client

MOD: Moving Objects Database

MLS: Mobile Locations Systems

MANET: Mobile and Ad Hoc Networks

PDA:	Personal Digital Assistant
PCS:	Personal Communications Services Networks
RA:	Registration Area
RW:	Random Walk
SA:	Service Area
WLAN:	Wireless Local Area Network
WAP:	Wireless Architecture Protocols

Chapter 1

INTRODUCTION

The ubiquity of pervasive mobile networks combined with the capability to support large databases promise new opportunities for mobile applications. However, in order to efficiently support these applications, mobile platforms are required to cache portions of the available data so that access over relative slow communication channels can get accelerated and communication disruptions may be minimized.

One of the most popular mobile applications is the provisioning of services using location information known as location-based services (LBS). LBS use the location information of mobile users to provide them with relevant information based on their geographical position [1, 6, 11]. Through LBS, mobile clients can access location sensitive information, such as traffic reports, hotel information and weather forecasting.

Location-aware dynamic caching and prefetching computing is a widely used technique used to improve performance in mobile computing environments [3, 9]. Prior research on data caching [12, 14] and prefetching [28, 11] has focused on the development of solutions that allow applications to run in wireless environments using proxy-based techniques. Even though most of these approaches are semantic-based, they do not provide adequate support for representing the context of a user (i.e., the interpreted human intention.). Although the context may be treated implicitly, it is still very crucial to data management in a mobile environment, since potentially context-sensitive and highly personalized information can be disseminated to mobile users. Even though the location information has been used as a key field of the user's query context, not enough attention has been paid to the other query fields (predicates) which define the user's information context. The characteristics of the mobile environments and the mobile user needs for dynamic provisioning of information content make most previous caching and prefetching techniques unsuitable.

This dissertation aims to solve this problem by presenting a novel data caching mechanism for pervasive computing environments that can adapt dynamically to a mobile user's context. In the dissertation we design and develop a conceptual model and context aware protocols for wireless data caching management. The presented replacement policy uses the validity of the data fetched from the server and the neighboring locations to decide which of the cache entries is less likely to be needed in the future, and therefore a good candidate for eviction when cache space is needed. The context aware driven prefetching algorithm exploits the query context to effectively guide the prefetching process. The query context is defined using a mobile user's movement pattern and requested information context.

In this chapter the principles of data caching, with a particular emphasis on cache management and prefetching, are first presented, then the goals, issues and assumptions of this dissertation are elaborated. Finally this chapter concludes with the scope and the outline of this dissertation.

1.1 Background and Preliminaries

1.1.1 Data Caching Overview

Data caching in mobile environments can support many applications and many levels of information systems design enhancing performance without significantly increasing the cost and complication of the overall information system design. Caching can occur at least at three different locations: (i) the server, (ii) the proxy server, and (iii) the client. Of these, server caching is used not a means to reduce delays, but as an effective way to reduce the demand for clients' connections on a single server. Enterprise applications where data caching is applied include the traditional client-server distributed environment, the web and the much newer mobile and wireless environments. Figure 1.1, shows a taxonomy of the main data caching applications, data caching techniques and the cache main processes discussed in this chapter. In a web environment, proxy servers deal with aggregate user demand. Caches are commonly located at the client and at proxy servers [62], which are found within a user's intranet. Client data caching is particularly important in mobile computing, because it can reduce wireless communication and has been shown to be an effective way to handle the above problems [18, 29, 14].

In traditional legacy systems a predictive caching system makes optimal caching decisions regarding the caching of the most important files, based on the types of user interaction with these files [4]. Through the concept of semantic distances the distance between files and directories is measured and quantified. The sequence in which the files are accessed can also be taken into account. One such approach is found in the SEER system, which was implemented as a modification to the UNIX

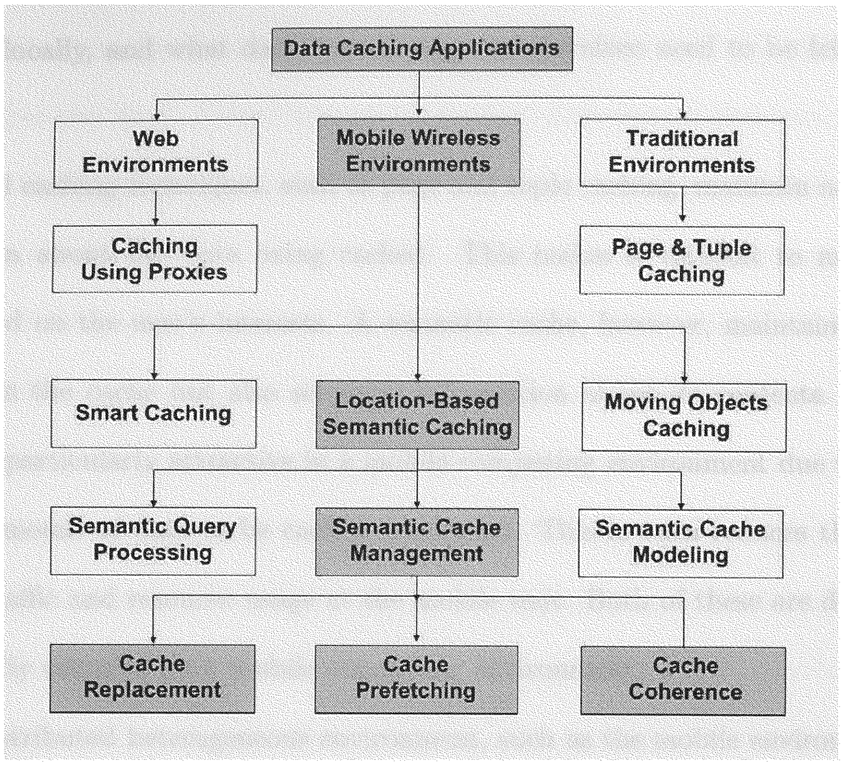


Figure 1.1: A data caching taxonomy.

kernel and a collection of user-level processes to calculate semantic relations among the files. These were logged by the kernel in a trace buffer and examined later to make optimal caching decisions.

While the traditional data dissemination systems use physical information to organize data and caching techniques based on page and tuple caching [13, 5], newer data caching techniques maintain the associated answers (data items) of previous queries as well as their meaning (purpose). This form of information description is called *semantic description*, and makes it possible to reason about and derive knowledge from the given description. A semantic cache was defined in [7] composed of a set of cached items, which are called semantic segments. During a new query processing the semantic description is used to determine what percentage of the query can be answered locally, and what data are missing and therefore need to be fetched from the server.

Typical caching techniques, such as page and tuple caching, maintain no semantic information about the data being cached. This makes it difficult to manage the cache based on the user's interests. A semantic cache, however, maintains not only the data in the cache but also semantic information about its contents. Semantic caching is particularly attractive in a mobile computing environment due to the fact that the amount of data to be cached is reduced. This in turn reduces the wireless network traffic and resource usage at the mobile unit. Both of these are desirable in a potentially resource poor mobile computing environment.

In a distributed heterogeneous environment, such as the mobile environment, the semantic data caching paradigm is an important technique for improving the performance of wireless data dissemination systems. Semantic caching is by nature an ideal cache scheme for Location Dependent Data (LDD) applications due to the following reasons. First the semantic caching is built on the semantic locality among queries, which perfectly fits the LDD applications where semantic rather than temporal or spatial locality is exhibited [5, 7]. Secondly, continuous LDD queries can be incrementally processed by semantic caching. With each successive request, a much smaller trimmed LDD query is processed at the server side and only the differences are transmitted over the wireless link. Thirdly, semantic caching makes cache management more flexible.

Another data caching technique in a mobile environment called *object caching* relates to mobile objects or mobile clients who are on the move. These mobile objects need to be constantly provided with the relevant data they need depending on their location. This brought forward the problem of keeping track of the mobile objects and therefore the need of databases to support moving objects and data structures to support their functioning. These moving objects are represented in a Database for Moving Objects (DOMINO) containing the moving objects and their location [30].

Object caching techniques have demonstrated that discrete representations could be mapped into data structures that can be used in a DBMS environment providing a precise base for the implementation of spatio (space predicate) - temporal (time predicate) extension package to be added to a suitable extensible architecture. These moving object databases have to be continuously updated and queried. The update problem needs to determine when the location of a moving object has to be updated. These moving objects will generate random queries depending on their position. A critical set of capabilities such as the support for spatial and temporal information have to be built on top of existing databases in order to support moving objects databases [17]. Additionally, a new data model needs also to be introduced to check the location update policies and their performance.

1.1.2 Location Dependent Data Queries

A location dependent (LD) query is a query which is processed on location dependent data, and whose results depend on the location criteria explicitly or implicitly specified. Moreover, the result may change as the user changes location. An example of these queries may involve a traveler who gets different gas station information as s/he moves to different locations. Furthermore, since the same question is asked continuously, there may be a large degree of overlap in the results of consecutive queries Clearly, continuously evaluating a query from scratch would be very inefficient. If we keep a cache on the mobile unit, part of the new query result could be obtained locally. By doing this, the wireless network traffic is reduced, and the system performance is improved as well. To dynamically adapt to the query access pattern, the cached contents are required to move as the mobile unit moves.

Mobile clients querying static objects create queries like: "Tell me where is the nearest gas station?" and "Where is the nearest restaurant?" which are popular queries in real-world location applications. In general the mobile clients submitting this kind of queries are mobile and the data objects are fixed. The main challenge of this type of queries is how to get the location of the clients and how to guarantee the validation of the results when the client keeps moving during the query evaluation process.

Queries such as "Report all the available hospitals within a 500-meter radius" are an extension of this type of query. A LD query becomes more difficult to answer when it is submitted as a continuous query. For example, a client in a moving car may submit the query: "Tell me the room rate of all the hotels within a 500 meter radius from me" continuously in order to find a cheap hotel. Since the client keeps moving, the query result becomes time-sensitive in that each result corresponds to one particular position and has a valid duration because of location dependency. The representation of this duration and how to transmit it to the client are the major focuses of Continuous Queries (CQ). Sistla *et al.* [17], employed a tuple to bind the valid time duration of the query result.

Of course, the LDD queries listed above can also contain querying about location independent attributes, such as: "Tell me the nearest restaurant providing Chinese food." Since these queries can be broken down into two parts: one for location-dependent information and the other for location-independent attributes, we only consider queries on location-dependent information as the others can be retrieved by traditional query-processing methods. In particular, this dissertation focuses on the continuous type of queries initiated by mobile users querying static objects.

1.1.3 Cache Management Processes

Regardless of the cache techniques described in the previous section and the organization model being used the following data caching main issues (shown in Figure 1.1) make the design of client cache management for wireless data disseminations a challenge.

1. *Cache replacement*: Cache replacement refers to the process that takes place when the cache becomes full and old data items (objects) must be removed to make space for new (newly referenced) ones. An effective cache replacement policy is needed to decide which cache entries will remain and which will be evicted to make space for new data, since the cache is inevitably small to hold all the data referenced and, thus, some data must be evicted when new data is brought in. It is obvious that the cache will perform best if the data evicted is that which is least likely to be referenced again.

2. *Cache prefetching policy (also called cache hoarding)*: Cache prefetching automatically preloads automatically data values into the cache for possible future access requests. Prefetching in association with replacement manages the client's cache, providing an effective technique to handle information provisioning for users with limited resources and frequent disconnections.

3. *Cache coherence strategy*: Cached results from previous queries may become invalid when the mobile client moves from one location to another, or even when the objects residing at the database server are updated. A cache coherence strategy usually involves cache invalidation and update schemes to invalidate and up-date an outdated cached entry. An invalidation policy maintains data consistency between the client cache and the server. This is achieved by the origin server notifying clients if a cached object has changed. Another approach for cache coherence is cache validation. With cache validation, the clients verify the validity of their cached objects with the origin server. A detailed work in cache coherence control is presented in [29].

1.2 Motivation

In the real world, as mobile users change locations, they most likely tend to request the same data item a few times until switching to another one. For example considering the case of the mobile user asking questions to find the nearest hotel. 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. Then at time t_2 and location $cell\_id = x_2$, s/he asks query Q_2: "Give me the names and vacancy information for hotels within 5 miles which charge up to $100".

Affinity can be defined as the preference towards a particular database data item. Once the mobile client has demonstrated affinity towards one particular data item type (i.e., in this case hotel), 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

Table 1.1: Example of dynamic semantic cache index

S	S_R	S_A	S_P	S_{PT}	S_L	S_{CV}	S_{TS}
S_1	Hotel	Hname	$(L_{x1} - 5 \leq h_{xposition} \leq L_{x1} + 5) \wedge (L_{y1} - 5 \leq h_{yposition} \leq L_{y+5})$	2	L_{x1}, L_{y1}	110010	t_1
S_2	Rest/nt	Rname Type	$(L_{x2} - 5 \leq r_{xposition} \leq L_{x2} + 5) \wedge (L_{y2} - 5 \leq r_{yposition} \leq L_{y2} + 5) \wedge (6:00pm \leq Work - hours \leq 9:00pm)$	5	L_{x2}, L_{y2}	010111	t_2
S_3	Hotel	Vacan Hname	$(L_{x3} - 5 \leq h_{xposition} \leq L_{x3} + 5) \wedge (L_{y3} - 5 \leq h_{yposition} \leq L_{y3} + 5) \wedge (price \leq \$100)$	8	L_{x3}, L_{y3}	110010	t_3

movement pattern and the query pattern need to be considered to further improve the effectiveness of a cache management strategy. The presented data caching strategy can be used to incorporate dynamic attributes in existing data models and capabilities to be added to existing query processing systems to deal with dynamic attributes.

Next, observe that the number of moving objects in the database may be very large (e.g., in big cities with millions of inhabitants). Thus, for performance considerations, in answering LD queries examining the location of each moving object in the database should be avoided. In other words, it is preferable to index the location attribute. Therefore, the location granularity will be of great importance. Using a symbolic model based on cell granularity for location identification instead of the (x, y) geographical location models (as seen in the example shown in Tables 1.1 and 1.2) used so far is a better approach. The problem with a straightforward use of spatial indexing for location binding is that the continuous change of the locations implies that the spatial index has to be continuously updated. This is clearly an unacceptable solution that can be mitigated with the cell granularity symbolic location model adopted by this research.

Finally maybe the best motivation is that an increased in size cache can satisfy the expectations of more functionality at the client site (suggested by this research). Additional knowledge regarding the future location and query pattern of the mobile client can be included and it would be great to have available at the client. True Relational Data Base Management Systems (RDBMS), may now fit into 100K-150K of memory (e.g., DB2 Everywhere by IBM, Oracle Lite and Sybase's Ultra Lite databases). In implementing a small size RDBMS for the client, the symbolic model stresses the representation of relationships between logical entities rather their precise coordinates and is more suitable for LBS at a semantic level [21]. The fact that a cache can easily incorporate the functionality of a small database, can provide additional motivation for the use of the symbolic location model based on cell granularity.

1.3 Statement of the Problem

Traditional distributed database techniques cannot efficiently support queries in mobile computing environment. Consequently, query processing in mobile database, which is conducted by some fixed hosts and several mobile hosts, has emerged as an issue of growing importance [6]. A mobile database can be recognized as a distributed database that supports mobile computing. In general mobile wireless networks [3], there are two sets of entities: Base Stations (BSs) and hosts. The hosts are either fixed or mobile (called MH). Fixed hosts communicate over the network with a fixed topology, while mobile hosts communicate with other hosts (mobile or fixed) via a wireless channel. With the development of mobile computing, another kind of wireless network, both the hosts and the BSs are mobile. Thus, query processing becomes much more complex than that in general wireless network [10].

How to optimize mobile queries, cache and replicate data and manage transactions are some of the key issues in mobile environments which are grouped under the query processing cache management process. Location binding has been used as a means to mitigate these issues. The result of an LD query is not only determined by the selection and a projection condition specified, but also is related with a given bound location. However, current research has adopted the geographical location which uses the x, y location binding for the query processing (as seen in the example shown in Tables 1.1 and 1.2).

A more effective and native to query/database processing location model is adopted by this research. This is the symbolic location model which is based on cell granularity. Collecting information for individual mobile client at a finer level (i.e., x, y coordinates) would incur significant overhead without a significant performance im-

Table 1.2: Query predicate augmentation by client

id	Q	Query Predicate description	Time
1	Q_1	$(L_{x1} - 5 \leq h_{xposition} \leq L_{x1} + 5) \wedge (L_{y1} - 5 \leq h_{yposition} \leq L_{y1} + 5)$	t_1
2	Q_2	$(L_{x2} - 5 \leq r_{xposition} \leq L_{x2} + 5) \wedge (L_{y2} - 5 \leq r_{yposition} \leq L_{y2} + 5)$ $\wedge(6:00pm \leq Work - hours \leq 9:00pm)$	t_2
3	Q_3	$(L_{x3} - 5 \leq h_{xposition} \leq L_{x3} + 5) \wedge (L_{y3} - 5 \leq h_{yposition} \leq L_{y3} + 5)$ $\wedge(price \leq \$100)$	t_3

provement. 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 presented prefetching technique.

In some cases, prefetching is only beneficial for latency reduction, but causes a cost (i.e., cost of bandwidth and memory space) increase with respect to no prefetching. Prior research has not provided adequate support for representing the context of a user. However both the valid scope distribution defined as the locations of the fetched data where the queried data item has valid answers and the user's query context can be used as prefetching filtering mechanisms.

Finally, a significant amount of research has been conducted in future locations prediction, however, interesting enough it has not been adequately used to improve the cache management at the mobile client.

1.4 Research Goals

Based on the discussions in the previous sections the overall goal of this dissertation is to provide a novel data caching mechanism and a prefetching strategy for pervasive computing environments that can adapt dynamically to a mobile user's context. None of the solutions proposed so far has incorporated the query context and future locations into cache management and prefetching. This demands the design of efficient data caching mechanisms in mobile and wireless environments based on the issues and challenges arising out of their inherent characteristics and applications. This dissertation's aim is twofold:

- Design a future location-aware cache replacement policy to manage a semantic cache. The proposed replacement policy uses the validity of the data fetched from the server and the neighboring locations to decide which of the cache entries is less likely to be needed in the future, and therefore a good victim for eviction when cache space is needed.
- Design a context-aware prefetching strategy that will explore the user's query context to predict the future query pattern on a cost effective manner. This will enable the formation and maintenance of a context-aware cache with data items of high benefit and at a low cost.

By meeting these goals we expect to improve the cache query processing process by using *cell_id* location binding information instead of the geographical x, y coordinates for the query augmentation. Additionally provide more intelligent caching techniques at the mobile client for better local decision making. This will reduce the client-server dependency and improve the network resources allocation. In view of the above issues

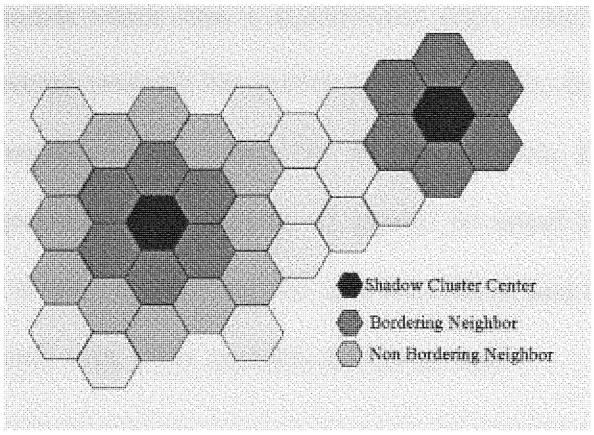


Figure 1.2: The shadow cluster concept([24]).

and challenges and keeping in mind the overall systems performance, we intend to incorporate the following main components into our research:

- Future location prediction. A future location prediction scheme is incorporated into the presented cache management strategy using the neighboring locations (cells). According to the shadow cluster concept (Figure 1.2), a cluster of cells based on the mobile user's current cell is formed. A bordering neighboring and the non-bordering neighboring group of cells are identified. Neighboring cells are most likely to be visited in the near future and therefore present a higher interest as fast as prefetching is concern. Remote cells are less likely to be visited in the near future and therefore are best candidates for replacement when additional cache space is needed. Even though the focus of this research is not in cells identifications, an algorithm has been included to identify the neighboring cells distribution and provide a number of other supplementary functions i.e., measuring the distance between cells.

- Symbolic location model based on cell granularity location modeling: Regarding the user's mobility the research has mainly emphasized the use of the geographical location models which uses a significantly high computational overhead. However most recent research has proven that a symbolic model is more appropriate for the mobile environments [21]. In this research, the symbolic model based on cell granularity is adapted. It should be noted here that the focus of this research is not on modeling the user's mobility, and the presented techniques can be used with other mobility model as well.

- The validity of the data concept (*valid scope distribution*) is adapted and used to derive a set of future locations of interest. The data item scope is defined as a set of cell where are the valid answers to the query and is maintained by the database server.

- 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*). Two query context criteria have to be designed based on the user's query context attributes and predicates. An algorithm is necessary to derive a set of candidate queries for prefetching using the user's previous history of the user's information content, then another algorithm uses two selection criteria to select the optimum query for prefetching based on cost. Cost is defined as the used transmission bandwidth and cache space.

1.5 The Dissertation Environment

There are several issues regarding the modeling of user movement, and the querying process that need to be defined before the study of the proposed algorithms.

- *Mobility model*: The random walk mobility model is assumed, however the proposed methods can easily be adapted to any other mobility model. The used mobility model includes random movement speed capabilities and is based on cell granularity several mobility models i.e., the Random Waypoint (RWP) mobility model and the one by Brinkhoff have recently attracted attention. This research has revealed that these models are still at a pretty nascent stage going through several modifications and used for ad hoc networks protocols simulation. Separate research streams exist on development and fine tuning of these models. For example "The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks" [86] has pointed out several drawbacks on using Waypoint mobility. Alternatively a directional mobility model is also used with some prefetching simulation experiments.

- *Query types*: The continuous LD query type is assumed and for local queries. It can be shown that any other type of query can be broken down and analyzed as if it was of continuous type. Local queries refer to the queries whose results are valid based on the current cell location of the mobile user. Non-local queries refer to the queries whose results are valid or are based in another cell which is not the mobile user's current location. For example "find the hotel names lower than $200 in West Palm Beach or in cell number "23". The presented approach can be expanded to include the non-local queries.

- *Mobile user's location*: Determining location it is assumed that MTs are equipped with a mechanism to obtain its current location. This mechanism could either be tied to the tracking scheme used by the static network (to setup a communication path to the client) or be completely independent of it. For example, the client may determine its location via Global Positioning System (GPS) technology in terms of its absolute geographical coordinates. Alternatively, in a environment with predefined cells, i.e. areas of wireless coverage with a common access point to the fixed network, a client's location can only be pinned down to the granularity of a cell, (method adapted by this research), thereby defining its location relative to they fixed network. So, the term location has different connotations depending on the mechanism used, with a range of granularity from a few square miles in the case of macro cellular structure (as in a cellular/Advanced Mobile Phone System (AMPS) setting) to a cell size of the order of meters, e.g. infrared cells within a building, "An Infrared Network for Mobile Computers" [87].

- *Valid scope maintenance*: It is assumed that a client cache is symbolically organized and each cache entry contains a data item ID, the attached validity information if any, and a pointer pointing to the real data. Note that the cached segment has a valid scope attached to it as of the time stamp of the last time it was accessed. When a data item is updated at the server-side, the cached copy becomes obsolete. When an MT hand-offs to another cell, a cached data item may also become obsolete due to the change of client location, depending on its valid scope. This paper does not deal with cache invalidation issues [21].

1.6 Significance and Contributions

The outcome of this research in the area of caching in a mobile environment would be used to make mobile devices more efficient in terms of the usage of their cache memory and for a wide variety of new wireless applications. Thus mobile and wireless devices implementing any kind of memory to support their usage would be a target of this application. This research is, of course, not limited just to mobile devices; ad-hoc devices which face the problem of constant disconnections, would also be candidates for the use of this application. Any kind of distributed system where clients frequently disconnect from the servers could eventually use this application to improve the way data is cached.

The presented techniques considerably enhance the query predicate with more accurate location and movement information. Hence the outcome will be the strengthening of the query prefetching capabilities and therefore the local processing of a larger number of location dependent future queries. Regarding the query processing by the client and in particular in trying to generate the probe and remainder queries, the intelligent algorithm included, significantly improves this process by discriminating the cached tuples based on their future location binding information.

Finally, the performance evaluation results suggest possible capabilities of the proposed strategy in a larger design/application space, where more mobile clients can be used to see if they interface or help each other in a peer-to-peer computing architecture, i.e., if there is no data access via a BS or an MSS, ask your peer.

1.7 Methodology

The presented research in this dissertation involves the combination of model development, algorithm design, analysis, simulation and experimentation. Model development involves a critical assessment of the requirements and challenges of mobile environments. The data caching techniques have been designed to satisfy these challenges and needs with concentration to emerging technologies needs rather than the traditionally used methods.

The proposed design was evaluated using both a cost analysis model and simulation techniques. Even though simulation is not quite foolproof, and only implementation in a real environment can assure the effectiveness of the design, the performance results were mainly derived using simulation due to the complicated nature of the mobile environment and the lack of real life testing platforms. For the same reason there is no available simulator for mobile environments and therefore one had to be designed for the mobile computing environments.

Despite the recent surge in research activities in the mobile and wireless infrastructure software simulation remains the primary approach to evaluate the data caching performance, as it is fairly easy to implement and manipulate. The presented methods for mobile environments are simple and practical for implementation and operation. The testing parameters used are available in practical systems. In order to evaluate the trustworthiness and overhead of the presented methods, a combination of analysis and simulation was adopted. An obvious extension of the work will be to create a mobile environments test bed and evaluate the presented techniques with real-time data.

1.8 Targeted Applications

The new classes of applications for mobile computing environment namely, LBS are expected to create a much higher user density in the future mobile networks. Also, the updating and querying loads on the location databases are expected to be very heavy [37]. Furthermore, with the technological improvements in pervasive computing, it is expected that a large number of location-aware wireless devices will be available in the future. The network of consumers using Personal Digital Assistants (PDAs), tourists carrying on-line and position-aware cameras and wrist watches, and vehicles with computing and navigation equipment will give rise to a wide variety of new wireless applications. The list of the emerging wireless applications includes the following:

1. *LBS Applications:*

   - Tracking and dispatching mobile resources
   - Traffic coordination, and way-finding
   - Location-aware advertising Tourist services, and
   - Location-based games.

2. *Multimedia Caching.* One of the research streams of the LandMARC project has looked at the use of multimedia caching as an effective means of supporting intelligent load balancing and resource management.

3. *Data caching in Mobile Ad-hoc Networks (MANET).* This is an emerging area of research which presents a promising future application for data caching [88].

4. *Additionally emerging application.* A great interest for data caching has been created with the biomedical engineering, tele-health, medical information systems and business.

1.9 Scope of Dissertation

The scope of this dissertation is to develop a conceptual model for cache management techniques and prefetching that could be performed for mobile devices and is focused on the LDD continuous type of queries initiated by mobile clients and for fixed objects. The scope of this dissertation demands the design of a new strategy or the enhancing of the present algorithm for caching to use a semantic model to efficiently manage LDD as well as answer normal queries for general data.

Other issues that fall in the scope of this dissertation are the future location prediction and query pattern prediction. The client cache management includes client side configuration parameters, cache fetching, cache replacement and cache prefetching policies. Other issues such as neighboring cells identification are also addressed to a lesser extent, since the main thrust of the thesis is on the cache management mechanism used for the mobile devices. The performance of the cache models is examined through analysis and a detailed simulation study, in order to show its effectiveness in a mobile computing.

1.10 Outline of the Dissertation

The work reported in this dissertation focuses on developing a novel future location prediction replacement policy and a context-aware prefetching strategy for the emerging location-depended computing paradigm. In Chapter 2, a survey of the major efforts conducted in recent years by the research community in the replacement and prefetching cache management issues is presented. This chapter also includes previous work in the associated areas of semantic caching and future locations prediction.

Chapter 3 first discuss existing data caching architecture for the pervasive computing environments including the semantic cache description and the use of the shadow cluster and the valid scope concepts for data caching. Next the design rationale of the cache replacement policy and its associates components is discussed. The last part of Chapter 3 discusses and compares our work with other protocols.

The presented Context-Aware Prefetching Strategy (CAPS) is presented in Chapter 4. The main issues of prefetching are discussed and the design of two algorithms used as context filtering mechanism is examined. The first to form a list of candidate queries for prefetching based on the user's current query context and the second algorithm to select the optimum query for prefetching. Next, the design of the prefetching cost model based on bandwidth allocation is presented followed by the numerical analysis and the performance results.

Chapter 5 presents the cache organization overview and the design of the simulator which was used to simulate the presented data caching techniques and prefetching method and derive the performance comparison results. A number of key simulation pseudo-code modules are included, such as the random movement control module, the cache manager module and the LRU, FAR and FLA replacement policy modules.

Chapter 6 summarizes the work of this dissertation, discusses the contributions it makes and the impact of the results to mobile computing. The last section in this chapter discusses suggestions for future research directions.

Finally, the appendices chapter, first, derives the formulas used by the neighboring cells distribution algorithm, i.e., the distance among cells formula, and the formulas used for the cell number and the reference number. Second, the transition probabilities evaluation is shown. Finally, in the performance statistical comparison appendix, we use the simulator's output and apply sampling techniques to construct confidence intervals (region) at specified confidence levels (i.e., 90% and 95 %), to compare the cache hit ratio of the proposed cache management strategies (CAPS and FLA) with the industry standard semantic cache management policy FAR and the baseline LRU protocol.

Chapter 2

RELATED WORK

Many researchers have advocated the use of caching and prefetching to reduce latency and improve the overall performance mainly in the traditional client-server distributed environments and the web. There are five other research areas that address issues discussed in this dissertation:

- Data Caching Techniques.
- Modelling the User's Mobility.
- Future Location Prediction.
- Cache Replacement, and
- Cache Prefetching.

2.1 Modeling Moving Objects

A formal data model Moving Object Spatiotemporal(or MOST for short) to represent moving objects in a database system was introduced in [17]. They treated the position of a moving object as a dynamic database attribute, and expressed it with three sub-attributes, namely value, update time and function, where the function indicates how the value changes over time. In [17], an effort was made to represent the temporal development of spatial entities in certain data types such as moving point or moving region. Moving Object Databases (MOD) have to be continuously updated and queried and the update problem needs to determine when the location of a moving object has to be updated. In addition, these moving objects generate random queries depending on their position. A critical set of capabilities such as support for spatial and temporal information, etc have to be built on top of existing databases in order to support moving objects databases [72]. The problem of how to index moving objects in a database is investigated in [71] and [73]. A straightforward use of spatial indexing is inefficient and infeasible since the spatial index has to be continuously updated when the objects are continuously moving.

In [7] a new semantic caching technique was proposed. A formal model, which treats location dependent data as database spatial replicas tightly coupled with specific data regions, is presented. Then based on this model, research issues such as query processing and the geographical model used for location binding are explored. The main advantage of geometrical location models, is their compatibility across heterogeneous systems. However, because of the considerable cost and the complexity involved in providing accurate fine-grained location information, the cost/performance ratio of geometric models might not be promising for a large number of applications such as the nearby restaurant example.

In this work, a symbolic location model is used, where the cell id is the unit of location granularity. Every cached item has an attached semantic description based on a cell granularity. In most of the emerging mobile applications the cell id number will be the preferable granularity. This approach differs from the previous research in semantic caching in that the knowledge of future location information is added as an extra rule for filtering information to be cached.

2.2 Future Location Prediction

The knowledge of the location adds an extra rule for filtering information to be cached, but its usability has been limited. Future location information can be used with the replacement policy. This particular technique offers a starting point for caching. In fact, when a mobile client is turned on, the only useful data available is its location. Hence, the caching system will start caching information steered by its location, and in the case this approach is feasible.

Zheng and Lee [12] have proposed to construct a *Voronoi Diagram* (VD) on the data objects to serve as an index for them. A VD defines, for each data object, the region within which the object is the nearest point to any mobile within that region, i.e., defined by the base station. Based on the VD, a semantic scheme that records a cached item as well as its valid range is proposed. The VD method even if, is a suitable approach to find the nearest-neighbor, it is not a real location prediction scheme and is seldom used in real applications because of the expensive maintenance of the structure when updating occurs. The *Dead Reckoning* schemes calculate where an object might be, instead of explicit updates. Other conventional methods on future location prediction are based on the use of historical movement patterns of the subscriber to calculate his possible future location. One method is based on a table with possible reference locations and on the probability that the MT is located in there in a deterministic period of time uses the time criteria to build the table index. A second method stores the historical movement pattern in a database and compares the recent states with these movement tracks in the database to find the one which samples the actual states (States dependent approach). One method to combine the time and the location information is to use the prediction characteristics of some neural network types, or use the stochastic prediction approach adapted by this research .

Levine *et al.* proposed the use of the *Shadow Cluster Concept* in [24] (Figure 1.2) for future location prediction and used it for resource allocation in Asynchronous Transmission Mode (ATM)-based wireless networks. The fundamental idea of the shadow cluster concept is that every MT with an active wireless connection exerts an influence upon the cells in the vicinity of its current location and along its direction of travel. As an active MT travels to other cells, the region of influence also moves, following the MT to its new location. The base stations (and their cells) currently being influenced are said to form a shadow cluster, because the region of influence follows the movements of the active mobile client like a shadow.

Using the shadow cluster model a matrix of transition probabilities was proposed, 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 as k increases, sometimes making the simulation of such system 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. In addition the shadow cluster model may be suitable for random walk only, in direction-based movement, the cells behind the user should not be considered for high access probability in the near future.

2.3 Cache Replacement

Cache replacement policies for the wireless broadcast environment were studied only in push-based broadcasts schemes [15, 16]. Furthermore, the previous studies are based on a number of unrealistic assumptions, such as, fixed data sizes, no updates, or no disconnections. Acharya *et al.* [15] proposed a cache replacement policy called PIX, in which the data item with the minimum value of p/x was evicted for replacement, where p is the item's access probability and x is its broadcast frequency. Thus, an evicted item either has a low access probability or a short retrieval delay.

The commonly used cache replacement strategies such as such as LRU, MRU and LFU are built on temporal locality. The least recently used (LRU) policy is an implementation of a probability-based policy. A probability-based policy replaces the data with the least access probability (evicts the object that has not been accessed for the longest time). LRU is unaware of any semantic relationships among queries. Aa a result, a significant number of queries that land in the cold region of the relation and therefore are not likely to be accessed in the near future, they will still stay in the cache until they age out of the LRU chain. The LRU policy works well when most recently referenced objects are most likely to be referenced again in the near future. However, there should be more factors to be considered in wireless communication systems when researching data caching replacement.

An alternative to using the recency information for determining replacement values is to use the *semantic distance.* In this approach, the data that is farther away from the client's current location is removed during replacement. In [5], Shaul *et al.* further utilize the semantic locality in semantic caching replacement. For each cached semantic region, a replacement value is assigned to represent the semantic distance between the “center of gravity” of that region and the “center of gravity” of the most recent query. With this distance function, semantic regions that are semantically “closer” to the current query are less likely to be discarded. The rationale used here is to predict the future use patterns for the segments by examining their semantic relationships with the current query. However, the calculation of the semantic distances has a number of issues associated with. First, the center of each region must be determined. Second, there is a need to use estimated weights which are difficult to determine most of the time. In a more recent research, Zheng *et al.* [13], proposed the data distance and valid scope area as important factors for the data caching replacement policy. According to their analysis, a promising cache replacement policy should choose its replacement data with a low access probability, a small valid scope area, and a long distance if the data distance is also an influential factor.

A different replacement policy is used in [14], where the cached items to be evicted are selected according to the access probabilities which are predicted by observing the data access history. In several studies on LDD caching, data distance-based cache replacement policies [5] utilize semantic locality in semantic caching replacement. For each cached semantic region, a replacement value is assigned to represent the semantic distance between the center of gravity of that region and the center of gravity of the most recent query. With this distance function, semantic regions that are semantically “closer” to the current query are less likely to be discarded. The rationale used here is to predict the future use patterns for the cached entries by examining their semantic relationships with the current query. A significant body of research on semantic cache strategies has used the distance and the direction of movement to determine replacement candidates to be evicted from the client's cache. This body of work implies the use of future location prediction in this process.

A mobility model that represents the moving behavior of mobile users is formally defined by Ren *et al.* in [7] for LDD queries. Based on this mobility model, a LDD semantic cache replacement policy was developed, named a Furthest Away Replacement (FAR). FAR chooses for replacement those segments which are not in the moving direction and are furthest away from the user. A future location is anticipated based on the current *user's direction* and *speed.* FAR implies future location prediction based on the tangent velocity. However, this approach is effective only within a short time interval, and therefore a great and frequent number of calculations is needed throughout and during the cache replacement decision making process. Additionally, for a small number of cells FAR would face a great difficulty to determine which of the previous locations is the furthest away. Furthermore, sudden changes in the direction of the MT make the FAR policy almost impractical. Xu *et al.* [20] proposed a cache replacement policy called Stretch Access-rate Inverse Update-frequency (SAIU), where the influence of data item sizes, data retrieval delays, data access probabilities and update frequencies are considered.

2.4 Cache Prefetching

Current research on prefetching is based on the tangent velocity approach, which is effective only within a short time interval and has a high cost for the continuous geometric estimations. Future location information can be used together with the prefetching strategy, a technique that offers a starting point for caching. In fact, when a mobile client is turned on, the only useful data available is its location. Hence, the caching system will start caching information steered by its location, and in this case this approach is the only feasible one. Prefetching is not at all a new concept. Since the very early days of microcomputer technology [58] caching and soon thereafter prefetching or preloading were integral parts of processors and file systems. Another early idea for file prefetching was to utilize application hints that specify future file accesses. This deterministic prefetching was explored by Patterson *et al.* [67].

At around the same time, the SEER project was born at UCLA [4]. SEER allow disconnected operations on mobile computers using automated hoarding. Following on the earlier work, SEER was extended to provide automated hoarding for mobile computers without the need of user intervention [4]. Whenever an imminent disconnection is realized by the system, a decision on the list of critical files is decided by SEER depending on the user accesses on the files, the semantic locality of the files and the sequence of these accesses. Then, the files are hoarded into the client system which is about to be disconnected. This hoarding (prefetching) is usually done periodically to ensure the availability of hoarded data in the case of involuntary disconnections.

With the arrival of mobile computing, the anticipation of data or file accesses is getting more and more crucial. *et al.* [2], were among the first to work out an intelligent file-hoarding tool. The tool automatically detects a user's file access pattern and hoards the files to present them in a convenient form at disconnection time. Hoarding provides only data that is stored while connected to the network. Hence, it is a scheme designed to increase the likelihood that a mobile client is able to continue working during periods of total disconnection from file servers. Prefetching on the other hand is mainly concerned with improving performance. The file server is assumed to be accessible, although the network connectivity may be weak. Although prefetching attempts to avoid cache misses, a reasonable amount of misses may occur. A simple consequence of a cache miss is a user based reactive information query and thus slower data access. Many of today's mobile information systems count on prefetching to improve the quality of their service.

Projects like the ones described in [82], [79] and [68] are partially or entirely based on WLAN or other high-bandwidth data access. Their approaches are based on so-called info-stations or hot-spotted areas. The idea is to provide the mobile clients with data at specific locations that provide access to WLAN or other high-bandwidth infrastructure. The difficulty lies in the prediction of the data needed on the way from one info-station to another, i.e. on the way to the next WLAN access point. The drawback of these algorithms is related to the hoarding problem. Data that is not present on the mobile devices can either not be accessed at all or has to be downloaded from scratch over the WAN [79].

The work of Pensone *et al.* [11] examines the effectiveness of prefetching policies in the location-aware mobile information services. It considered only the analytical evaluation of prefetching based on a Markov model, and did not consider how to effectively confine the prefetching information to reduce the communication costs as a whole. Most recently Park *et al.* [28] 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 Δt. 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 [7, 6, 13, 26].

The *geographical mobility models* inherently use continuous calculations of the tangent velocity, which is proven to have a considerably high processing overhead [33].

In summary, most of the existing methods are aimed at finding the most probable cell [35]. 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. there is a need for next-cell prediction.

2.5 Previous Work Summary

Despite this body of research and its very important findings that to improve wireless network communications, there are still barriers that preclude the full utilization of the mobile computing capabilities. Most of the existing methods are aimed at finding the most probable cell [35, 54, 55]. In addition, there are no existing studies that have considered both the future location prediction and the validity of the data fetched from the server (the *valid scope distribution*) concept. Evaluations show that there exists no "best" replacement or prefetching strategy. Depending on the workload, different strategies can give the best result. Function-based strategies can be made adaptive by changing the weighting parameters.

An area that hasn't been researched sufficiently is the combination of cache replacement and prefetching with cache coherence. Traditionally, cache replacement and prefetching are treated as separate topics. Nevertheless, there is a relationship between these main cache management mechanisms because a cache should not return stale web objects. In order to limit the amount of prefetched information, current research has used *geographical location models* to focus solely on the mobile user's movement pattern [26, 7]. The geographical mobility models inherently use continuous calculations of the tangent velocity, which is proven to have considerable high processing overhead [33, 34]. In this dissertation the location binding is based on cell location granularity versus the x, y coordinates used by most of the previous research. The comparison between the location models and the justification of the one adapted by this research is given in chapters three and four.

Chapter 3

A FUTURE LOCATION-PREDICTION REPLACEMENT POLICY

Data caching performance depends heavily on the replacement policy being used for the cache management. However, future location ambiguity, limited client resources and frequent client disconnections make cache management a challenge. The presented Future Location-Aware (FLA) replacement policy [42] uses the validity of the data fetched from the server and the neighboring locations to decide which of the cache entries is less likely to be needed in the future, and therefore provide a good victim for eviction when cache space is needed. For better efficiency the overall replacement granularity is dynamically achieved along three levels: the ring, the cell and the data item. Simulation study of the presented approach shows that it outperforms both the LRU and FAR schemes, where only temporal locality is considered. Moreover, the presented scheme is easier to implement than the industry standard policies.

3.1 Introduction

A cache replacement policy determines which data item(s) should be deleted from the cache when the cache does not have enough free space to accommodate a new item [52]. The choice of a particular cache replacement policy can have a significant impact on global network traffic and on local resource utilization by making the best use of available resources, including memory space and network bandwidth. The cache will perform best if the data evicted is that which is least likely to be referenced again in the near future. In effect, an effective replacement policy would enable a larger volume of location dependent data (LDD) queries to be processed locally, thus maximizing network bandwidth and increasing the overall cache management performance. Location dependent data is the data whose value is determined by the location to which it is related. Moreover, the result of a query may change as the user randomly changes location while repeating the same query (continuous LDD query).

To adapt dynamically to the query access pattern, the cached data items are required to be reorganized, according to the mobile user's movement pattern and query pattern. This observation motivates the development of cache replacement strategies using movement pattern and location, with location being a key field of the mobile user's context. Traditional cache replacement policies (i.e., LRU) do not perform well in wireless data dissemination [21]. In these policies, the access probability is considered the most important factor that affects cache performance. A probability-based policy is used to replace the data with the least access probability. The LRU policy removes the data items which have not been accessed for the longest period of time. This policy works well in workloads which exhibit strong temporal locality (i.e., recency of reference). One of the main challenges of data caching in a mobile computing environment is how to predict the future location of the MT, and how to use this information to effectively manage its cache. A substantial amount of research has focused on location-prediction based on tangent velocity, which is effective only within a short time interval [5, 7]. However, there have been insufficient studies in applying future location-prediction and valid scope (locations where the queried data has valid answers) in deriving an effective cache replacement policy.

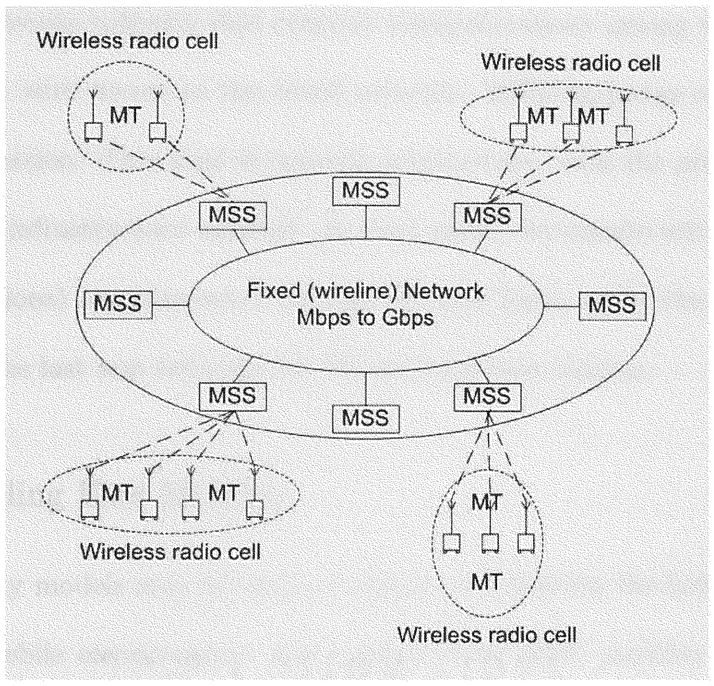


Figure 3.1: A mobile architecture.

3.2 Wireless Infrastructure Description

In a mobile architecture (Figure 3.1), 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 trough 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.

3.2.1 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 [43, 44]. In this work, the RW model is used primarily with the directional (DIR) movement model included as a secondary model. However, the adapted model and the pre-fetching technique are independent of the mobility model selected in the experimental section. The motion models considered (analogous to motion models already presented in previous works [24, 25, 26, 27, 31, 32] serve just as examples of practical application, other models could be adopted as well.

Movement is considered in an area divided into adjacent "cells" (Figure 3.2). Each cell consists of all the locations that share common information that must be provided by the service as an 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 γ, or remains in the same cell with probability $1-\gamma$. 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, the distance between two cells is defined as the minimum number of cells that must be traversed to pass from one cell to the other, and as ring k the set of all cells whose distance from a given cell is equal to k. The prefetching-strategy outlined above, for a given k (circle of radius k) prefetches all the information associated to rings *0, 1, 2, ...* k around the starting position. 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).

3.2.2 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 ex-

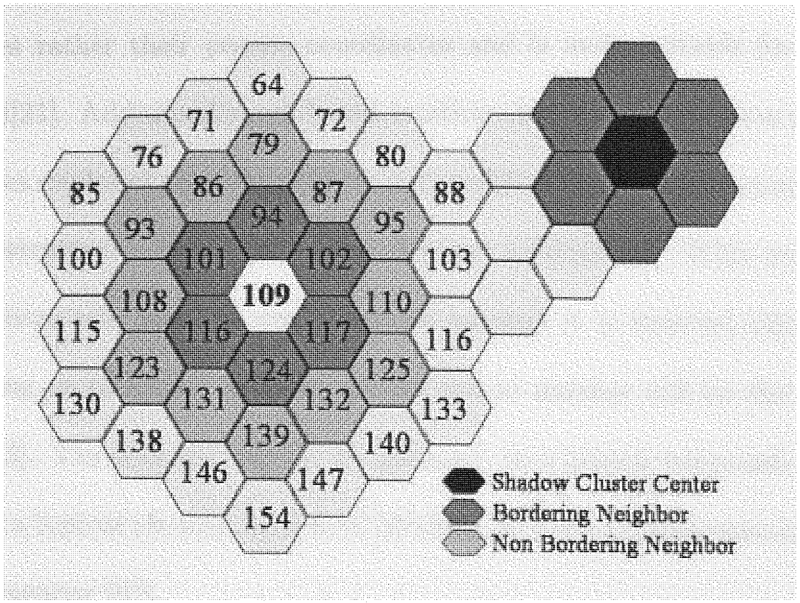


Figure 3.2: The shadow cluster concept-neighboring cells distribution.

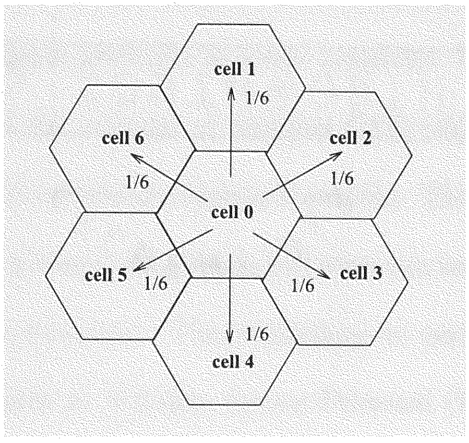


Figure 3.3: Random walk model - next cell probability distribution.

ample, 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 presented prefetching technique. Additionally, the fact that a cache can easily incorporate the functionality of a small database , 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 their precise coordinates and is more suitable for LBS at a semantic level[21]. 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[19]. With recent developments in micro-cell, pico-cell and nano-cell systems it is believed that for most of the emerging mobile LBS applications the cell id number will be the preferable granularity [13]. The average cell size in diameter, in a typical micro-cell system, is 100m-1km, in a typical pico-cell system is 10-100m, and in a typical nano-cell system, is 1-10m in diameter [37].

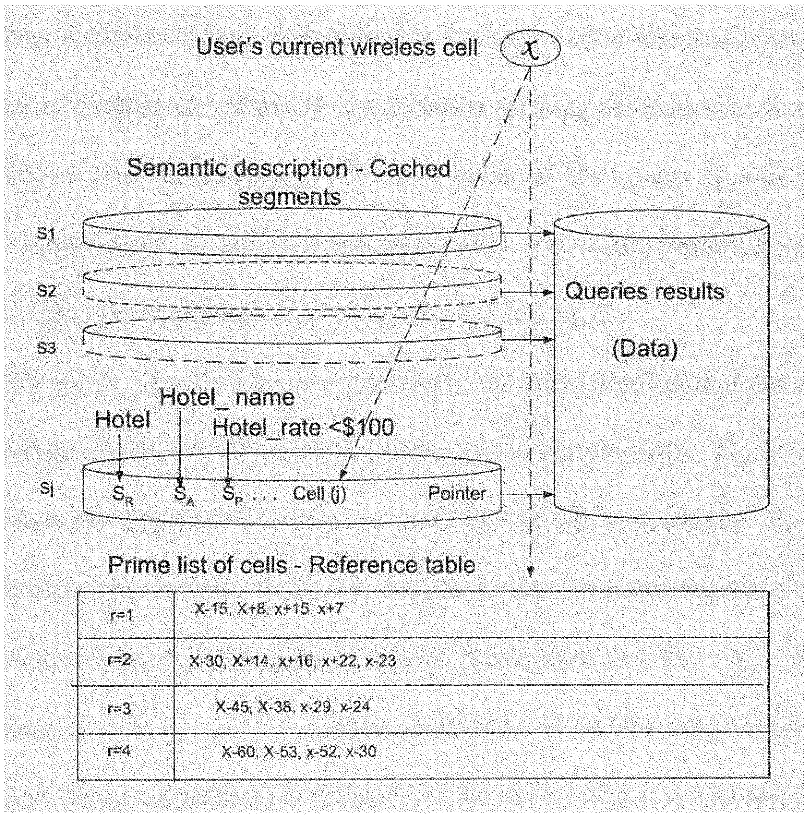
Determining the exact location requires satellite technology which is still not widely available in cell phone networks in many countries. Cellular communications themselves have spawned the concept of Assisted GPS/(AGPS) where the network assists the GPS receiver to perform its various functions. The most demanding AGPS environments tend to be in inner-cities where cell sizes can be limited to a few kilometers in radius (www.gpsworld.com). The technology to perform Mobile Positioning (MP) under a protocol such as Wireless Access Protocol (WAP) is being exploited to develop a variety of value-added services through a WAP mobile phone or mobile station. A cell-shape based MP methodology using WAP has been proposed, which can apply to the existing Global System for Mobile Communications (GSM)/Digital Converter System (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 the 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 present it to different instant messaging 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 U.S. 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 [85]. 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.

3.2.3 Prediction Level

Basically, the more cells or the larger the areas considered in the prediction, the better the approximation that can be reached, however the required computations will be increased. According to the concept of shadow cluster (Figure 3.2), the influenced cells are a group of cells surrounding the cell in which the MT is residing. Thus, the current cell is used as the starting point, thus being the center of a shadow cluster. The vicinity of the current cell can be denoted according to its distance away from the center cell. If a cell is adjacent to the current cell, then it is in the first layer of the current cell. The cells adjacent to the first-layer cells form the second layer of the current cell. If the estimated cells cover only the cells of the first layer, then first-level prediction is used. Similarly, the second-level prediction is associated with both first-and second-layer cells. When a LDD query is submitted for execution, the location where the mobile client is currently visiting instead of its whereabouts within the cell is needed to run the replacement process.


User's current wireless cell
x
Semantic description - Cached segments
S1
S2
S3
Queries results
(Data)
Hotel_ name
Hotel
Hotel_rate <$100
Sj
S_R
S_A
S_P . . . Cell (j)
Pointer
Prime list of cells - Reference table

r=1	X-15, X+8, x+15, x+7
r=2	X-30, X+14, x+16, x+22, x-23
r=3	X-45, X-38, x-29, x-24
r=4	X-60, X-53, x-52, x-30

Figure 3.4: Semantic cache structure and the prime list of cells.

3.2.4 The Client's Semantic Cache Description

In this section a brief presentation of the semantic cache is given using a logical model (Figure 3.4) that uses the cell granularity for location identification instead of the (x, y) geographical mobility models used in previous work [6, 26, 7]. 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[7]. 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 $S =< S_R, S_A, S_{pt}, S_L, S_{ts} >$.

In this definition, S_R and S_A are respectively the base relation and the attributes in S. S_{pt} represents the link to the first page that stores the segment. S_{ts} is the timescale indicating when the segment was last accessed by the cache manager. $S_P = P_1 \vee P_2 \vee .... \vee P_m$ indicates the criteria which the tuples in the semantic segment S satisfy. In the S_P equation, P_j is a conjunction of simple predicates, i.e., $P_j = b_{j1} \wedge b_{j2} \wedge .... \wedge b_{jl}$. Each b_{ji}, where $i = 1, 2, ..., l$ is a simple predicate. Π is the project operation that lists the subset (Π_{S_A}) of attributes defined by the query and σ is the select operation, that selects the tuples (σ_{S_P}) to satisfy the predicates requested by the query.

Example 3.2.4: *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:

Table 3.1: State of the Cache Example

Q_i	S_i	S_A	S_P	S_{ts}	S_L	S_{pt}
Q_1	S_1	$Hname$	x_1	t_1	x_1	2
Q_1	S_2	$Rname$	$x_2 \wedge$	t_2	x_2	5
		$Type$	$(6:00pm \leq schedule \leq 9:00pm)$			
Q_1	S_3	$Hname$	$x_3 \wedge (price \leq 100)$	t_3	x_3	8
		$Vacancy$				

"Give me the names and types of the restaurants within 5 miles that are open from 6:00pm to 9:00pm", 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. Table 3.1, gives a snapshot of the cache state with the three cached segments S_1, S_2 and S_3.

3.3 Query Modelling and Augmentation

This work focuses on continuous local query types. Local queries refer to the LDD queries where results are valid based on the current cell location of the mobile user. Non-local queries refer to the queries where results are valid or are based in another cell which is not the current MT location. Even though this paper focuses on local queries, it can be expanded to include non-local queries. It is noted that the definition of a new query language is not necessary as the existing query language is adapted for LDD queries presented in the literature [47] (SELECT *projections* FROM *set-of-*

objects WHERE *Boolean-conditions*). Where *projections* is the list of attributes to retrieve from the selected data items (objects), the *set-of-attributes* is a list of data items (object classes) interesting to the query. *Boolean-conditions* used to select data items by restricting their attributes values i.e., by demanding the satisfaction of certain *location-dependent* constraints.

Consider a database $D = \{R_i, 1 \leq i \leq n\}$, where R_i is a relation for $i = 1, 2, 3, ...n$. Furthermore, let A_{R_i} stand for the attributes set of R_i, and let $A = \cup A_{R_i}$ represent the attribute set of the whole database. A location-dependent query Q can be expressed by a tuple arrangement, $Q =< Q_R, Q_A, Q_P, Q_L >$, or by the *relational-algebra expression* $Q = \Pi_{Q_{R_1}, Q_{R_2}, Q_{R_3}, ..., Q_{R_n}}(\sigma_{\mathrm{Q_P}}(R_1 \times R_2 \times ... \times R_n))$. In this definition, $Q_R \in D$, $Q_A \subseteq A_{Q_R}$ is the set of selected attributes, Q_P is the set of query predicates, and Q_L is the current cell number. Π is the project operation that lists the subset (Π_{Q_A}) of projected attributes defined by the query. σ is the select operation (*query predicates*), that selects the tuples $\sigma_{\mathrm{Q_P}}$, to satisfy the predicates requested by the query. Ans(Q) denotes the retrieved tuples (query results).

Location binding: The result of an LDD query is not only determined by the selection $(\sigma_{\mathrm{Q_{P'}}})$ and the projection (Π_{Q_A})conditions specified, but it is also related to the given location binding, that means $Ans(Q) = \Pi_{Q_A}\sigma_{\mathrm{Q_{P'}}}(Q_R)$, where $Q_{P'} = LocBind(Q_P, Q_L)$. Our strategy is to treat the cell location information as a normal condition. Therefore, the first step of LDD query processing is to perform the corresponding predicate location binding. Throughout this paper, it is assumed that a location-dependent query is bound to a mobile user's current location unless explicitly specified. The assumed binding granularity is the projected location based on the movement of the MT. After the predicate location binding, an LDD query is converted into a normal database query, which can be processed from the local cache using strategies similar to the ones proposed in [7].

3.4 The Cache Replacement Policy Components

A mobility-based semantic cache replacement policy was first presented in [5] utilizing the semantic value function. Ren *et al.* [7] by using the location information attached to each segment, made it more efficient. The presented method improves this future location awareness approach and add the query affinity factor. Additionally, the presented method mitigates both the accuracy of the probability scores assigned to the future predicted cell and calculations overhead [26, 24], by trimming the candidate list of neighboring cells using the *item valid scope distribution.* It is noted that the presented approach is also an improvement to the great work done in replacement by Baihua *et al* [13]. Next, in this section, a detailed explanation is given of the key components of the replacement policy and related algorithms. In addition, the query pattern method and the replacement granularity , which are used with the presented replacement policy, and the overall replacement model and algorithm.

3.4.1 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* ($v_{i,j}$). 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 [13, 21, 42]. 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 happens because every data server supports different service areas. Hence, a LDD query will produce different answers from each data server.

Example 3.4.1: 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 3.2.4) 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 $U_i = u_{i,A} + u_{i,B} = \{1, 2, 3, 4\}$.

3.4.2 Movement Pattern - Future Cells for Replacement

In order to form and maintain a cache with high value cached items, the first consideration is to predict the cells that will contain the most likely future query results. Assuming that a cluster of cells is composed of k rings, where $k = 1, 2$, and 3 denote rings 1, 2 and 3 respectively. To be consistent throughout this paper a cluster configuration of three rings ($k = 3$) of cells is used ($R_{x,k=1}$,$R_{x,k=2}$, and $R_{x,k=3}$), where x is the innermost cells (center cell) of the cluster and quite often denoted by $R_{x,k=0} = x$. Figure 3.2 depicts a cluster of three rings for a total of $3k^2 + 3k + 1 = 37$ cells. Using the underlying geometry, the number of cells in the k^{th} *ring*, denoted by N_{R_k}, is given by Equation 3.1, and the number of cells in a cluster configuration with k rings, denoted by N_{C_k}, is given by Equation 3.2. Notice that $\mathrm{N}_{R_k} = N_{C_{k+1}} - N_{C_k}$.

$$\mathrm{N}_{R_k} = \begin{cases} 1, & k = 0 \\ 6\mathrm{k}, & k > 1 \end{cases} \tag{3.1}$$

$$\mathrm{N}_{C_k} = 3 \times k(k+1) + 1 = 3k^2 + 3k + 1 \tag{3.2}$$

Next, two groups of cells are defined, the Bordering Neighboring Group (BNC) of cells and the Non-Bordering Neighboring (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. Nevertheless, 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, a detailed calculation of the probability distribution can be provided 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 from these cells. To mitigate this problem and also to improve the accuracy of the future location prediction, the presented prefetching policy uses the data item valid scope distribution (U_i) as a masking operator applied on the *neighboring list* of cells (C_i) to derive a subset of cells called the *prime list* denoted by P_i, where $P_i = C_i \wedge U_i$. Next, example 3.4.2, uses a cluster of cells composed of three rings and an arbitrary scope distribution U_i to demonstrate how the prime list of cells is derived.

Table 3.2: Neighboring Valid Cells Identification

$R_{x,k=1}$	$x-15$	$x-8$	$x-7$	$x+7$	$x+8$	$x+15$
$R_{x,k=2}$	$x-30$	$x-23$	$x-22$	$x-16$	$x-14$	$x-1$
	$x+1$	$x+14$	$x+16$	$x+22$	$x+23$	$x+30$
$R_{x,k=3}$	$x-45$	$x-38$	$x-37$	$x-31$	$x-29$	$x-24$
	$x-21$	$x-9$	$x-6$	$x+6$	$x+9$	$x+21$
	$x+24$	$x+29$	$x+31$	$x+37$	$x+38$	$x+45$

Example 3.4.2: Neighboring cells identification. Table 3.2 shows the displacement number which is added to the user's current cell number x to identify the next cell the MT moves to. This number depends on the mobile user's current cell x and the movement direction selected. The values of $k = 1, 2$ and 3 denote rings 1, 2 and 3 respectively. The displacement is inherited by the geometrical grid and is used to form a 15x15 grid of cells. Figure 3.5 depicts a 7x7=49 cells portion of the total grid. The MT may move from the current cell (lets assume that $x = 109$) to cells located at N, NE, SE, S, SW, NW (Figure 3.6). If the next selected direction is N, the next step will be to cell 94 ($x - 15 = 109 - 15 = 94$).

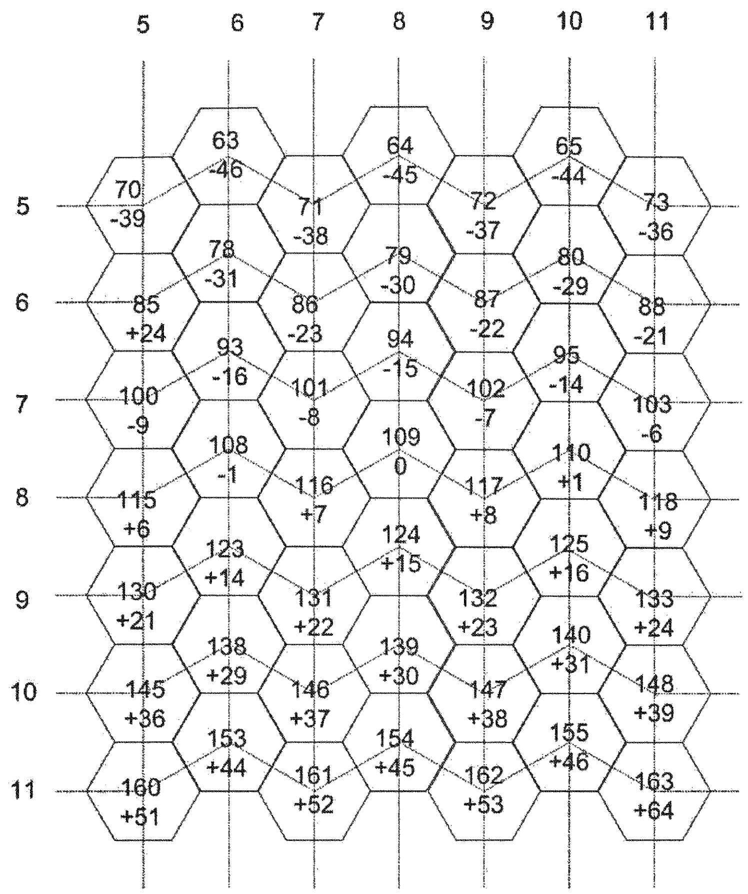


Figure 3.5: A four ring cluster (7x7 grid) of valid cells.

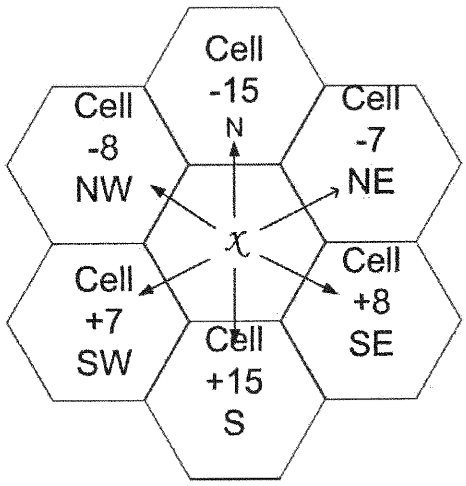


Figure 3.6: Next ring neighboring cells identification model.

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 S direction, the next cell will be 124 ($x + 15 = 109 + 15 = 124$). 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 scope distributions of the data items are maintained and updated periodically by the database server. The following equations show the calculations for the neighboring and prime lists of cells.

$U_i = \{x + 8, x + 15, x + 7, x + 14, x + 16, x + 22,$

$x - 45, x - 38, x - 29, x - 24, x - 60, x - 53$

$x - 52, x - 39, x - 44\}$

$C_i = R_{x,k=1} + R_{x,k=2} + R_{x,k=3}$

$P_i = C_i \wedge U_i = \{x + 8, x + 15, x + 7, x + 14, x + 16,$

x+22,$x - 45, x - 38, x - 29, x - 24\}$

$P_i = \{117, 124, 116, 123, 125, 131, 64, 71, 80, 85\}$

$P_i = \{117, 124, 116\}_{k=1} + \{123, 125, 131\}_{k=2} + \{64, 71, 80, 85\}_{k=3}$

Table 3.3: Example of Prime Index Table

Ring number	Prime list of cells for random cell($x = 109$)
$k = 1$	117, 124, 116
$k = 2$	123, 125, 131
$k = 3$	64, 71, 80, 85

3.4.3 The Future Location Transition Probabilities

So far a list that contains the neighboring cells has been formed. However, only the ones which are part of the data item scope distribution are of interest to the presented techniques. The question is what is the probability distribution of the primary list of cells based on the MT's current location. To find this distribution an appropriate set of transition probabilities needs to be adapted. There is a great number of research on transition probabilities, some of which was presented in section two. In this section a random movement model is presented that can be used together with our replacement strategy.

During the replacement process the cache manager in order to efficiently locate the candidate segment(s), it uses the prime list of cells (Table 3.3) to examine the cached segments according to the future location transition probabilities between their bound *cell_id* k and the query's bound *cell_id* x initiated by the mobile user. In modelling users mobility various movement models can be used. Each mobility model comes with each own matrix of transition probabilities G. Liu *et al.* [63] have described a number of movement models such as regular daily and weekly movement patterns and they have used Markov Chain Model to derive the transition probabilities matrix $\mathbf{P}_{x,k}$. Levine *et al.* in [24], have also derived a transition probabilities matrix using the Markov Chain Model for their *Shadow Cluster Concept* movement model.

In Appendix B the transition probabilities are evaluated using the γ movement probability. γ can be considered as a measure of the mobile user's speed. A low value i.e., $02, 0.3$, indicates slow movement while γ=0.8 indicates fast movement. Then a_i, b_i and c_i, denote the probabilities that at the end of each time slot the MT moves to ring $i-1$(previous ring), $i+1$(next ring) or remains in the same ring i, respectively.

Algorithm 1 The Replacement Algorithm $FLA(C, X)$

```
/* $C$ denotes the LDD cache and $U_i$ denotes the data
item $(i)$ valid scope distribution. $r$ is the cluster size
$V_{min}$ is the needed cache space.
x denotes the mobile user's current cell_id
y denotes the cell_id the cached segment $s_j$ is
associated (bounded) with */

1:  INPUT:(x,r), OUTPUT: VictimsList V
2:  / Step 1. Form the Prime List $(P_i)$ of Cells
3:  $C_i = R_{x,k=1} + R_{x,k=2} + R_{x,k=3}, ..., +R_{x,k=r}$
4:  $P_i = C_i \wedge U_i$
5:  $P_i = \{P_{i,k=1}\} + \{P_{i,k=2}\} + \{P_{i,k=3}\}, ..., +\{P_{i,k=r}\}$
6:  V ← NULL
7:  / Step 2. Use the movement pattern to look for
8:  / segments associated with remote cells zooming
9:  / from $r^{th}$ ring to the $1^{st}$ neighboring ring
10: for {int k=r, k >0, k - -} do
11:     for {every segment seg (j) in C} do
12:         while {$V < V_{min}$ and $y \in P_{i,k}$} do
13:             V ← V + seg(j)    /discard $(s_j)$;
14:         end while
15:     end for
16: end for
17: / Step 3. If more cache space is needed use the query
18: / pattern factor to examine segment's data item type (j)
19: for {every segment seg (j) in C} do
20:     while {$V < V_{min}$ and $i \neq j$} do
21:         V ← V + seg(j)    / discard $(s_j)$
22:     end while
23: end for
24: return {free space is enough}
```

3.4.4 The Replacement Score Model

A cache replacement policy involves computing a utility function $\phi_i(t)$ for each cached data item S_i that can be potentially replaced, and then replacing (expelling) the ones from the cache (the so-called victims) that have either the minimum (replacement value) or the maximum (replacement score) utility function value, depending on the criterion for replacement. In the semantic region model of cache organization [5], each semantic region is assigned a replacement value, those items with the lowest values are chosen to be the replacement victims. In our case, the utility function is the replacement score calculated for each cached data item. Based on guidelines presented in the previous subsections. The presented Future Location-Aware (FLA) policy assigns a replacement score to each cached segment using first the mobility primary factor and then if more space is needed the secondary query pattern factor.

The movement pattern primary factor (m_i) is calculated by forming the prime list of cells and comparing the location bound (cell_id C_i) for each potential for replacement cached data item within the prime list. Most of the time, but not necessarily, the query's bound cell C_i will be the cell to which the mobile user is located when s/he initiates the query. Assuming a cluster of K rings ($k = 1, 2, 3$), the prime list will most likely have three groups of cells. In the Example 3.4.2 the prime list was shown to be formed by $(117, 124, 116)_{k=1}$, $(123, 125, 131)_{k=2}$ and $(64, 71, 80, 85)_{k=3}$ groups of cells. Depending to which group of cells (1,2,3) C_i belongs to, the mobility factor value towards the overall replacement score will be 1,2,3 respectively. The mobility factor highest value ($m_i = 4$) is assigned to a cached data item which is bound to a cell that is no longer part of the data item scope distribution.

The query pattern secondary factor is formally defined by the affinity (f_i) between the current query Q and each of the candidate segments (data items) for replacement. Affinity is the probability that the user asks for the same data item type in the consecutive queries. In a location-aware information service, 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. The answer to a continuous type of query (e.g., a moving car asking for hotels located within a radius of five miles) needs to be updated continuously. Hence, 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. It is possible that they could get sidetracked every now and then - those would be the times when they query for something else.

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. 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 may be used to further improve the effectiveness of a cache replacement policy. Three levels of affinity are considered as follows:

- Low affinity level, ($f_i = Low$) to denote the probability of asking the same data item is $p \leq 0.33$.
- Medium affinity level, ($f_i = Medium$) to denote the probability of asking the same data item is $0.33 \leq p \leq 0.66$.
- High affinity level, ($f_i = High$) to denote the probability of asking the same data item is $p \geq 0.66$.

Using the mobility and query pattern factors a replacement granularity is implemented. The replacement mechanism first looks for potential candidate data items for eviction associated with remote cells (i.e., data items whose cell bound is part of the prime list outer rings). Next if more space is needed, the cached data items bound to cell which are members of the first neighboring ring of cells are examined. Lastly, the replacement mechanism uses the highest granularity to search for additional replacement victims in the MT's current cell implementing the query pattern factor(query data item type affinity).

3.5 Replacement Performance Evaluation

In this section, the performance of the presented cache replacement policy, namely FLA, is evaluated using the simulation model described in the previous section. For performance comparison the "traditional" policy LRU and the semantic policy FAR are used. Additionally comparisons are made with the replacement policies PA and PAID proposed in [13]. The *cache hit ratio* (h) is used as the primary performance metric. This is because most of the other performance results can be derived from the cache hit ratio, which is the percentage of all requests that can be satisfied by searching the cache for a copy of the requested data item. Additionally, the response time is included as an alternative performance metric.

The mobile client mobility is patterned using the random walk model. Each iteration is simulated until the MT has completed a fixed number of movements. Each iteration uses a set of valid cells that are defined based on the current cell. The expected average hit ratio $h_{AVG} = \frac{\sum_{i=1}^{N} h_i}{N}$ is estimated for each replacement 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). The averaged results are compared for all competing replacement policies. Only the steady-state behavior is of interest. 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 the next sub-sections, a number of experiments is described to investigate the performance of the examined schemes focusing on the characteristics that could impact the replacement decisions, such as cache size, simulation time, affinity factor, data item types, query delay, valid scope size, and movement speed. Steadily falling costs of storage lead to caches of sizes large enough to hold requested objects of various sizes [48], therefore the object size factor (presented in detail in the Greedy Policy [53]) has not been included in this study .

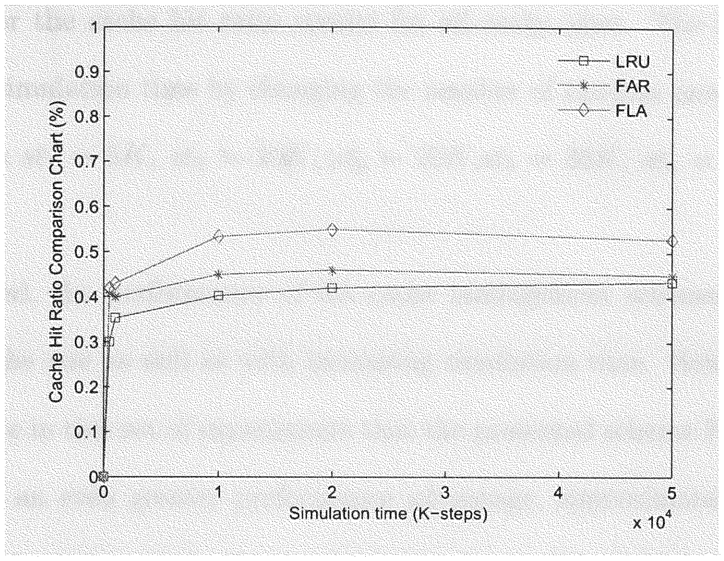


Figure 3.7: Cache hit ratio versus cache size - for various simulation times

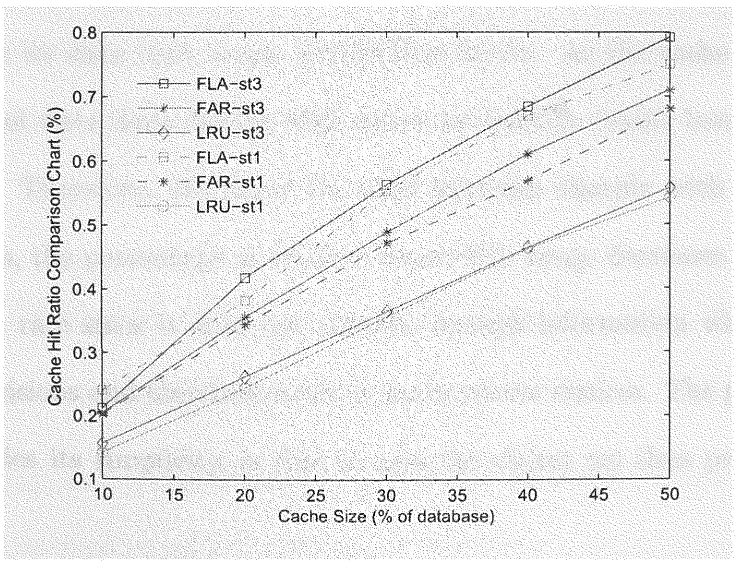


Figure 3.8: Impact of cache size for different simulation time

3.5.1 Impact of the Cache Size

Figure 3.7 depicts the cache hit ratio against the simulation time and Figure 3.8 depicts the cache hit ratio against the cache size variable values using a low (st1) and high (st5) value of simulation time. For each value of the simulation time value is averaged for the cache hit ratio results for all cache sizes. The experiments use an increased simulation time by changing the number of random movements for each iteration from $st_1 = 5K$, $st_2 = 10K$, $st_3 = 20K$,$st_4 = 30K$, $st_4 = 40K$ and $st_5 = 50K$.

As expected, the performance of the cache management schemes improves with increasing cache size as well as with increasing simulation time. However, it is interesting to notice in this set of experiments that the presented scheme FLA (Figure 3.8) demonstrated an even greater performance advantage, approximately $10 - 25\%$ for higher simulation time, while the standard direction policy (LRU) and the semantic replacement policy (FAR) improved by an approximate $5 - 10\%$. This is a result of FLA being able to adapt itself to the data items distribution in the neighboring cells, due mainly to its data item scope distribution factor. As the cache size increases, more important data items having high access probability (cache benefit) are stored in the cache. Therefore, the cache hit ratio increases sharply with the cache size increase. Thus, the percentage of wireless bandwidth usage decreases. LRU achieves the lowest hit rate since it does not consider enough information when making replacement decisions and therefore tends to make poorer choices. The positive feature of LRU, besides its simplicity, is that it ages the object set thus preventing cache pollution.

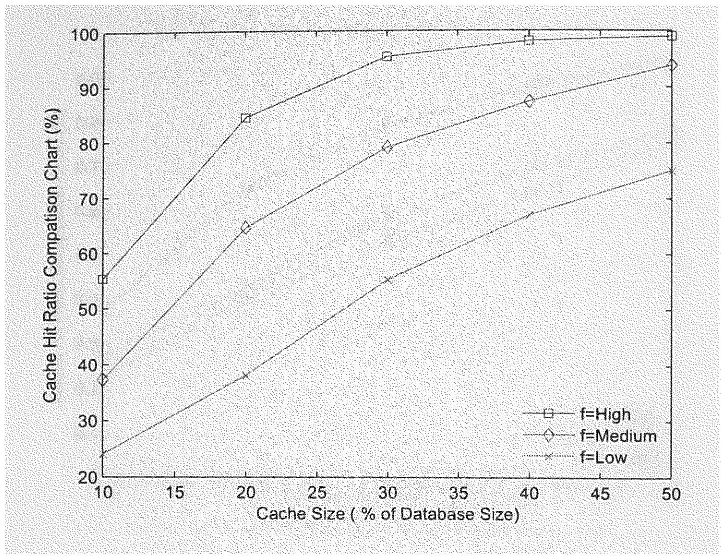


Figure 3.9: Changing the replacement query's pattern (Affinity factor) for FLA.

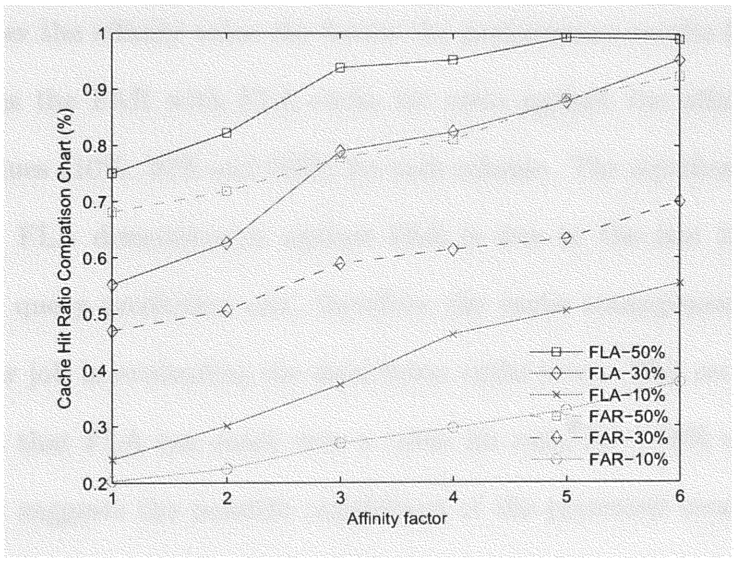


Figure 3.10: Affinity factor - Comparing FLA and FAR.

3.5.2 Impact of the Affinity Factor

While wandering, a mobile user may have an affinity (preference) towards certain item type (continuous type of query). The query pattern is simulated using the affinity factor (i.e., from low, $f_i = 1$, to high, $f_i = 6$). Figure 3.9 demonstrates, as expected,

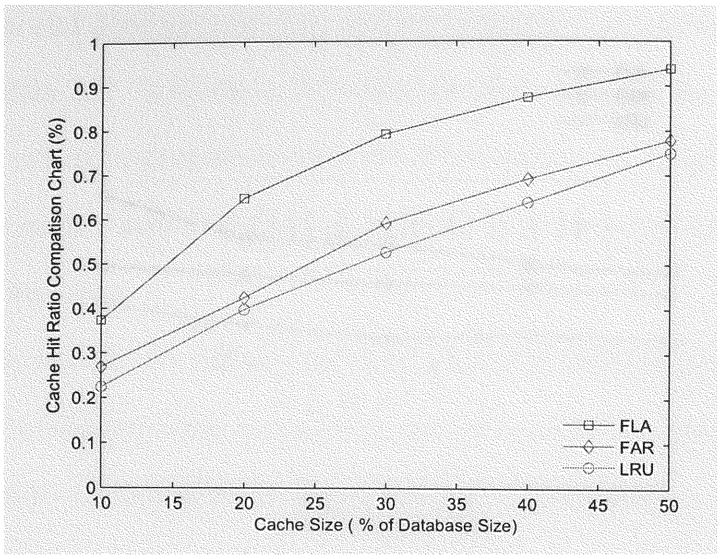


Figure 3.11: Cache hit ratio for FLA-FAR-LRU vs cache size.

that the higher the affinity value the better the performance results for FLA. Figure 3.10 compares the FAR with FLA cache hit ratio against the affinity using three cache size values (10%, 30% and 50%) for each scheme. The significant performance improvement FLA demonstrates against FAR is due to the fact there is a much better future query prediction and, therefore, the cache management strategy does a much better job in evaluating the data items cache score. This set of experiments demonstrates that FLA can reach very a cache hit ratio for a 50% cache size. This characteristic suggests the possible capabilities of the presented strategy in a larger design/application space, where more MTs can be used to see if they interface or help each other in a peer-to-peer computing architecture, i.e., if there is no data access via BS or MSS, ask your peer. Note that FLA is the only policy that uses the affinity factor, a fact that makes the comparisons with the other schemes not straightforward.

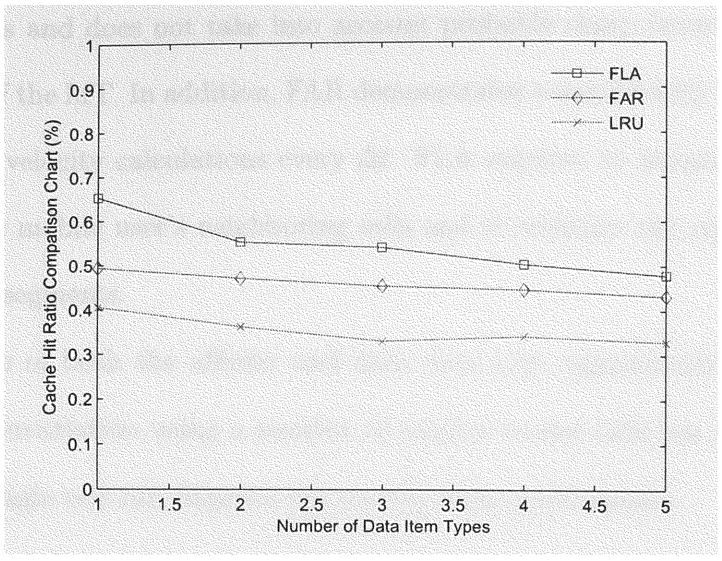


Figure 3.12: Cache hit ratio versus the data item types factor.

3.5.3 Data Item Types

In this set of experiments, the scalability of the presented replacement scheme is examined under various data item types. Figure 3.11 depicts the cache hit ratio versus the cache size using three data item types. Figure 3.12 demonstrates the cache hit ratio performance for five data item types. For each value of the data item type factor the hit ratio values are summarized for all cache sizes. As expected, the performance of the replacement schemes decreases when the number of data types increases. However, the FLA policy still shows a 5% to 25% higher performance compared to FAR, depending on the number of different types of data items used in the database.

An even higher improvement is shown compared to the LRU policy. This happens because FLA explores its future location capabilities in a 360^0 global scope, while FAR 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 take into account probable sharp turns (really random movements) of the MT. In addition, FAR demonstrates a significantly higher overhead of continuous velocity calculations every Δt. FLA requires an insignificant overhead to identify the mobile user's neighboring cells and to estimate the replacement score of the cached segments.

The results of both the affinity and data item type experiments suggest that a cache size differentiation using a number of smaller caches (one per data item type) instead of a single one can improve the overall cache performance.

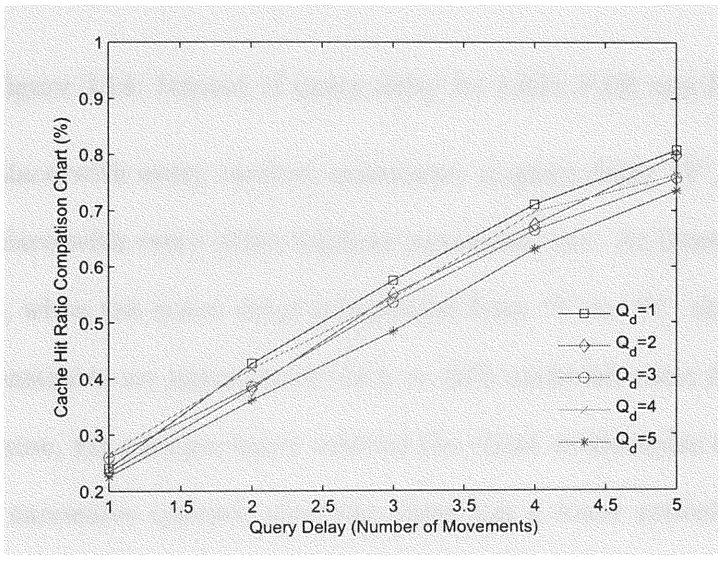


Figure 3.13: Impact of query delay on FLA.

3.5.4 Impact of the Query Delay

The query delay is the time interval between two consecutive client queries. It is noted that it is not necessary to query for an item each time a new movement is chosen. In this set of experiments the impact of the query delay is evaluated on the cache hit ratio under the contending replacement policies. A query delay of "1", means that the

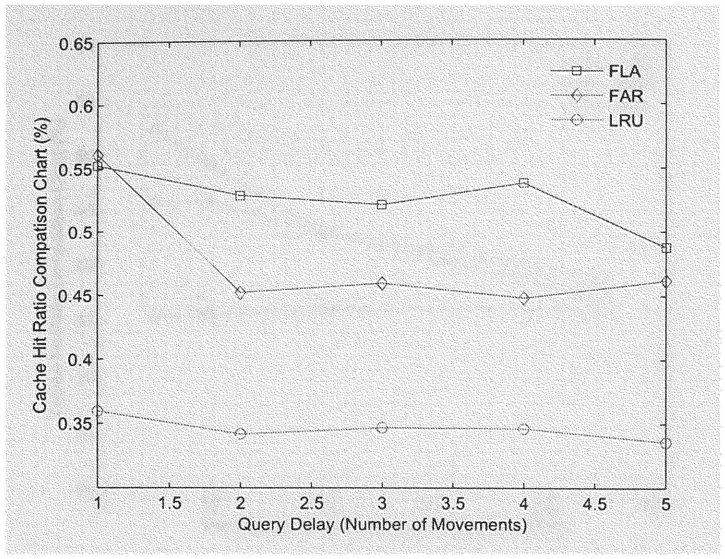


Figure 3.14: Impact of query delay for LRU, FAR and FLA.

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 in Figures 3.13 and 3.14, when the query delay is increased from "1" to "5", the FAR and LRU policies demonstrates an approximate 18.5 to 25% 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 presented strategy responds differently by displaying a smaller decrease in the cache hit ratio (only a 10% decrease) as the query delay increases from "1" to "5". This result is expected due to the fact that FLA makes replacement decisions using broader scope i.e., cell numbers within rings rather than continuous location (directions) estimations the competing schemes use.

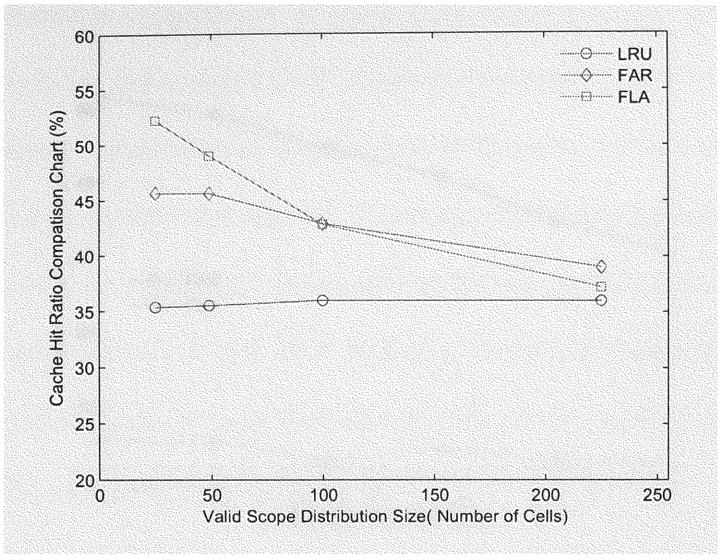


Figure 3.15: Performance results based on scope distribution size.

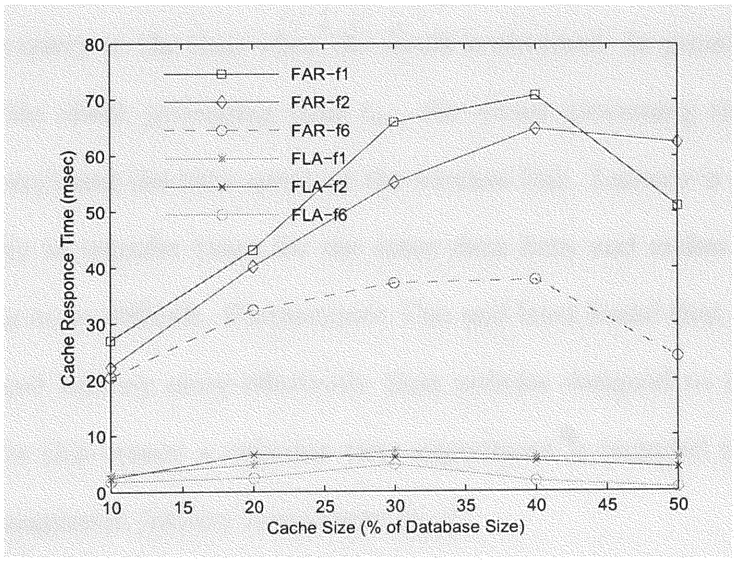


Figure 3.16: Cache response time for FAR and FLA versus cache size.

3.5.5 Mobile Cache Response Time

In a mobile environment, whatever allocation is chosen, users will experience some latency in getting the appropriate information when they cross the boundaries of the current information scope. The latency or response time, is the time from the sub-

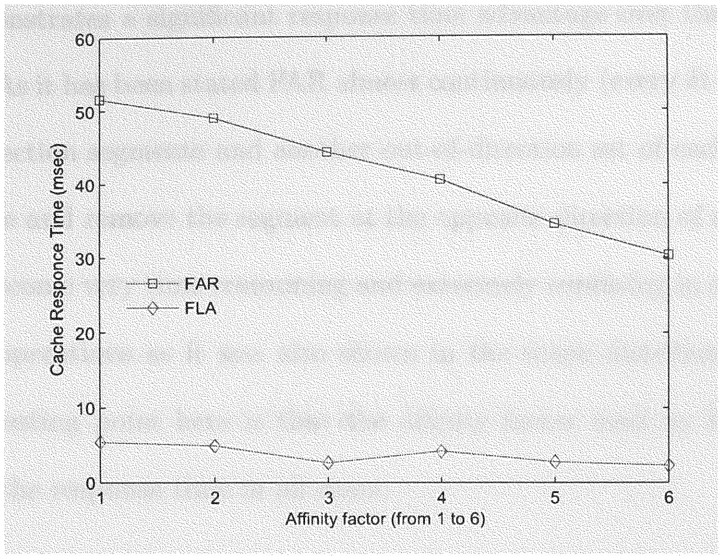


Figure 3.17: Cache response time for FAR and FLA versus affinity factor.

mission of the query to the time when the result is obtained. In general, this involves three parts: the client processing time t_{loc}, the server processing time t_{rem} (i.e., a server or a proxy) and the time spent on the wireless link. Latency is associated with high variability in transfer times for the same data item and makes cache management decisions more difficult. Furthermore, Cao and Irani found that maximizing the hit rate reduced latency more effectively than policies designed to reduce response times [53]. For this reason a response time experiment is included to examine only the cache management latency component (t_{loc}).

In this set of experiments FAR and FLA are compared. LRU has a much lower cache processing overhead due to its simplistic algorithm compared to the other two policies. Figure 3.16 depicts the cache average response time against the cache size which varies from 10 to 50% of the database size, using three different values for the affinity factor. Figure 3.17 depicts the cache response time average results over the five different cache sizes while the affinity factor value changes from low to high.

FLA demonstrates a significant response time advantage over the FAR, approximately 20%. As it has been stated FAR almost continuously (every δt time) calculates a set of in-direction segments and another out-of-direction set of cached segments in order to decide and remove the segment at the opposite direction of movement. This process can become very time consuming and extremely confusing in small to medium size areas of operations as it was also shown in the scope distribution experiment. Another interesting point here is that the affinity factor used by FLA can reduce substantially the response time in all cases.

3.6 Replacement Policy Conclusions

In this chapter, the cache replacement issues were investigated in a wireless data dissemination environment. Noticing that the query has valid answers only within a set of cells predefined for each data item (item valid scope), a novel approach based on the data validity to form a prime list of future cells was presented. The presented policy examines all cached items and forms a future location replacement score index, comparing the cell to which each cached item is bound with the prime list of cells. When cache space is needed, the replacement policy selects the candidates for replacement from the tail of the replacement score index. FLA is based on an efficient logical model that uses cell size granularity rather than the (x, y) geometrical model used by many other works.

The simplicity of our mobility model makes the prediction algorithm highly efficient and avoids making complex calculations which are unavoidable in stochastic models. Thus, the power required to process the cached segment and select the replacement candidate list used by the replacement policy is greatly reduced. FLA incorporates the scope distribution associated with the cached segments for increased accuracy of future location prediction and query pattern. Previous data caching research treats cache replacement and cache coherence as separate topics. The presented policy takes a modest first step towards the collaboration of these two cache management functions, by considering the valid scope distribution of the data items provided by the database server. Results show a substantial performance improvement over the existing LRU and FAR policies especially in a small to medium size of data items valid scope distribution and in higher movement speeds.

Previous data caching research treats the three main issues - coherence, replacement and prefetching individually. Studies have shown that, when combined with caching, prefetching can improve latency by up to 60%, while caching alone offers at best 26% latency improvement [66]. In the next chapter the prefetching strategy is presented in a collaborative approach with the replacement strategy.

Chapter 4

A CONTEXT-AWARE PREFETCHING STRATEGY

In a mobile wireless environment, the latency (time-delay) observed by a user before s/he receives up-to-date information may be high because of the limited available bandwidth. An efficient prefetching strategy must be tailored to the competing goals of keeping latency low (which requires more prefetching) and reducing resource waste in a mobile environment, which is characterized by scarce bandwidth and resource-poor user devices. This chapter presents a new Context-Aware Prefetching Strategy (CAPS) [41] and its components.

CAPS 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. An analytic model is used to compare the cost versus latency. Simulation results show that the presented strategy, using different levels of granularity, can greatly improve system performance in terms of the cache hit ratio.

4.1 Introduction

Previous research in data caching has focused on the use of the location information as the key field of the *user's query context*, but not enough attention has been paid to the other query fields (predicates) which define the user's information context. 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, CAPS 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 [5]. Ren *et al.*[7, 8] 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 both the neighboring cells and the current query information context factors are considered.

Based on these two factors, CAPS 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. Then, in order to identify data items with a 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 presented 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.

4.2 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

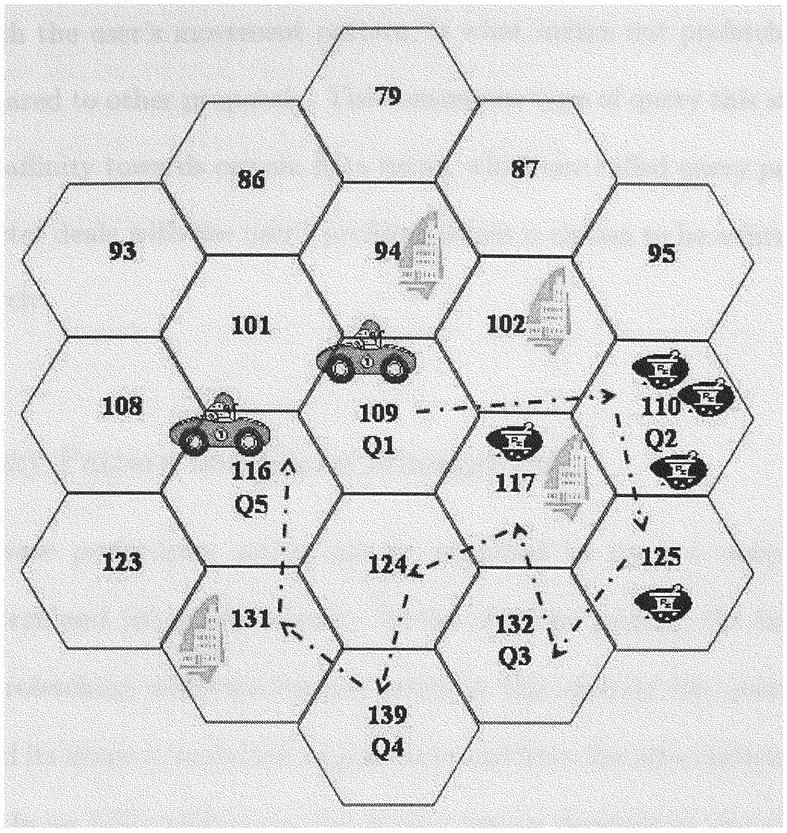


Figure 4.1: Roaming mobile user.

already used data in a cache. Prefetching, together with replacement, is 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.

In the previous chapter the user's movement pattern was presented, in this chapter the query pattern is explained in detail. A query pattern is a parameter which together, with the user's movement pattern, is what makes our prefetching strategy unique compared to other proposals. The continuous type of query this study focuses on shows an affinity towards certain data items, which are called query patterns. The third parameter deals with the user's profiling which is chosen to be a future direction of this research.

4.2.1 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, the future cells of interest to prefetching were determined, while in this chapter the query context is presented and its usage is explained to granulate and filter the information inside these chosen 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.

As discussed in the previous chapter 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

Algorithm 2 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
$LP$ denotes the list of past predicates for the relation
of $Q_i$, ordered by descending frequency. */

begin
Case 1. Add one frequently requested attribute at a time
for each attribute a in $Q_i$ do
    $Q_i$=add a to requested attributes of Q
end for
if $Q_i <> Q$ then
    add $Q_i$ to LQ
end if
Case 2. Add one frequently requested predicate at a time
for each predicate p in LP do
    $Q_i$=add p to predicates of Q
end for
if $Q_i <> Q$ then
    add $Q_i$ to LQ
end if
Case 3. Remove one predicate of Q at a time
for each predicate p in Q do
    $Q_i$=remove p in Q
    add $Q_i$ to LQ
end for
Case 4. Remove a predicate from Q and at the same time
add an attribute (combination of above cases)
for each predicate p in Q do
    for each attribute a in LA  do
        $Q_i$=remove p in Q
        $Q_i$=add a to requested attributes of Q
    end for
end for
add $Q_i$ to LQ
end
```

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 two main issues are identified: (i) how

Algorithm 3 Prefetching Overall Algorithm

```
/* $Q_i$ denotes the candidate queries for prefetching
$Q$ 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 ($Q_{iA} \subseteq Q_A$)
$\phi_2-$The tuple similarity criterion
indicates $Q_i$ has subset of tuples of Q ($Ans(Q_i) \subseteq Ans(Q)$). */

1: begin
2: Case 1. Form the Prime List of Cells (Prefetching Zone)
3: $P_i(prime) = C_i(candidate) \wedge U_i$
4: Case 2. Form a List of Candidate Queries
5: Use Sub-Algorithm 2 to form the list of candidate
6: queries $LQ$ for prefetching.
7: Step 3. Select the Optimum Query Context
8: Use the two criteria ($\phi_1$ and $\phi_2$) to compare $Q_i$ with $Q$
9: while cache space is available do
10:    Select queries $Q_i$ that satisfy both criteria
11:    Select queries $Q_i$ that satisfy one of the two criteria
12:    Select queries $Q_i$ that partially satisfy the two criteria
13: end while
14: Step 4. Augment Optimum Query and Prefetch
15: Use the PZ future cells to augment selected $Q_i -> Q'_i$
16: Prefetch using augmented queries $Q'_i$
17: end
```

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.* First, the query history is examined to find the most frequently requested attributes and query predicates, and then are subject to augment or alter the current query using this history. To limit the space of candidate queries only the following cases are considered. These cases 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)

Table 4.1: Query Candidates List for Prefetching

Q: "give me all the names of the Chinese restaurants at the current location in the medium price range"

- *Q*1– "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. and in the medium price range"
- *Q*2– "give me all the names and price ranges of the Chinese restaurants at the current location in the medium price range"
- *Q*3–"give me all the names and price ranges of the Chinese restaurants at the next location "
- *Q*4– "give me all the names of the Latin restaurants at the current location"
- *Q*5– "give me all the price ranges of all Chinese restaurants at the current location in the medium price range"
- *Q*6– "give me all the names of the Chinese restaurants at the current location"
- *Q*7– "give me all the names and price ranges of the Chinese restaurants at the current location"

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),

Table 4.2: Candidate Queries-Attributes and Predicates Description

	Relation: Restaurant	
Q_i	Attributes	Query predicates description
1	Rname	$p_{11} = current\_cell(x)$
	$(A_{R_{12}})$	$p_{12} = Type(Chinese)$
		$p_{13} = Schedule(6 - 10p.m.)$
		$p_{14} = Price(Medium)$
2	Rname, Price	$p_{11} = current\_cell(x)$
	$(A_{R_{12}}, A_{R_{15}})$	$p_{12} = Type(Chinese)$
		$p_{14} = Price(Medium)$
3	Rname	$p_{15} = future\_cell(k)$
	$(A_{R_{12}})$	$p_{11} = Type(Chinese)$
		$p_{14} = Price(Medium)$
4	Rname	$p_{11} = current\_cell(x)$
	$(A_{R_{12}})$	$p_{16} = Type(Latin)$
5	Price	$p_{11} = current\_cell(x)$
	$(A_{R_{15}})$	$p_{12} = Type(Chinese)$
		$p_{14} = Price(Medium)$
6	Rname	$p_{11} = current\_cell(x)$
	$(A_{R_{12}})$	$p_{12} = Type(Chinese)$
7	Rname, Price	$p_{11} = current\_cell(x)$
	$(A_{R_{12}}, A_{R_{15}})$	$p_{12} = Type(Chinese)$

and (iv) remove a predicate and add a requested attribute (combination of (i) and (iii)). Based on this described logic, the sub-algorithm (Algorithm 2) is

Table 4.3: Candidate Queries's Cost Description

Q_i	Q_{iA} vs Q_A (Selected Attributes)	$Ans(Q_i)$ vs $Ans(Q)$ (Result-Tuples)	Bandwidth and Space Savings	Query Changes
1	$Q_{1A} = Q_A$	$Ans(Q_1) \subseteq Ans(Q)$	Yes	Q_P
2	$Q_{2A} \supset Q_A$	$Ans(Q_2) \subseteq Ans(Q)$	Yes	Q_A more attributes
3	$Q_{3A} \supset Q_A$	$Ans(Q_3) \cap Ans(Q)$ $= NULL$	No	Q_A, Q_P, Q_L - new attributes and tuples
4	$Q_{4A} = Q_A$	$Ans(Q_4) \cap Ans(Q)$ $= NULL$	No	Q_P, Q_L new tuples
5	$Q_{5A} \cap Q_A$ =NULL	$Ans(Q_5) = Ans(Q)$	No	Q_A new attributes
6	$Q_{6A} = Q_A$	$Ans(Q_5) \supset Ans(Q)$	Yes	Q_P some new tuples
7	$Q_{7A} \supset Q_A$	$Ans(Q_6) \supset Ans(Q)$	Yes	Q_A, Q_P - some new attributes and tuples

designed to form a list of candidate queries LQ for prefetching. Table 4.1 lists part of the algorithm's output using query Q for input. Additionally, Table 4.2 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, the following observations can be made regarding the prefetching cost of the candidate queries list. Considering prefetching to be a form of caching for dynamically generated content [34], one can also consider prefetching along a number of granularity levels such as the cell, relation, attribute and item values ($Cell \rightarrow Data\_item \rightarrow Attribute \rightarrow Item\_Value$). In predicting the user's next query, the two criteria can be derived regarding the prefetching cost of the above listed candidate queries. The following two criteria are used to determine the benefit of prefetching a query:

(ϕ_1) *The attributes commonality criterion* is satisfied if the candidate to prefetch query Q_i, shares attributes with the current query Q.

(ϕ_2) *The tuple similarity criterion* is satisfied if the result tuples of Q_i, are a subset of the result tuples of Q.

The best candidates for prefetching are the queries Q_i that satisfy both criteria (ϕ_1, ϕ_2). A high affinity level is assigned to this query, i.e., $f = High$. If there is still cache space available, the queries that satisfy either one of the two criteria (ϕ_1, ϕ_2) are prefetched (medium affinity level, $f = Medium$). Finally, the queries that partially satisfy the two criteria can be used for prefetching (low affinity level $f = Low$). 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 [38, 39, 40] a compatible future location-aware replacement policy was described that is used here with the prefetching strategy when the cache is full. Table 4.3 lists the candidate queries using a bandwidth and space allocation cost taxonomy.

The overall steps for the presented prefetching strategy are shown in Algorithm 3. Using the above described criteria, it is noted 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 is expected to achieve a good prediction level for the next query. As a second best choice for prefetching query selection, queries Q_2 and Q_6 meet only one of the two criteria. Q_2 asks for a new attribute and Q_6 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_3, 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.

4.3 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. This section focuses on the bandwidth and uses it as the cost of prefetching the consumption of this scarce resource. The measure of bandwidth occupancy is represented by 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"*. P_{miss} is the probability of not finding the required information locally when an *out-of-scope* condition occurs (cache miss), while the probability of a cache hit is denoted by 1-P_{miss}.

During a cache miss, the information is retrieved from a remote source in t_{rem} time units (e.g., a server or a proxy server), and during a cache hit the information is retrieved locally in t_{loc} 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 factors that affect latency, such as the time needed to detect the out-of-scope condition, are not considered in this dissertation. 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 t_{rem}. 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 t_{rem} can be excessively high.

Prefetching is considered when a cache miss occurs. Clearly, if the main concern was to reduce the overall performance, then it would be recommended to simply increase the frequency of prefetching and the amount of prefetched data. A compromise is needed between two opposite goals - (i) to keep P_{miss} small and (ii) to optimize resource usage i.e., to determine an ideal number of cells to be used for prefetching. Therefore, a cost evaluation model needs to be defined to find the equilibrium between cost and performance.

The average occupation of the wireless link can be expressed as $W = IPC + TNM$. The $IPC = W_q + W_a N_0$ component, denotes the initial prefetching cost and $TNM = t_q b_k \sigma_k (W_q + N_i W_a)$ denotes the cost of the total number of misses. The expression for W can be written as follows:

$$W = (W_q + W_a N_0) + t_q b_k \sigma_k (W_q + W_a N_k), \tag{4.1}$$

where N_k is the number of cells for prefetching when the i^{th} cache miss occurs. For a standard random movement $N_k = 3k^2 + 3k + 1$. N_0 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 $b_k = \frac{2(k-1)+1}{6(k-1)}$, using the state diagram shown in Appendix 6.3, corresponds to the exit from the cluster covered by previously prefetched information and to the reconstruction of a new cluster.

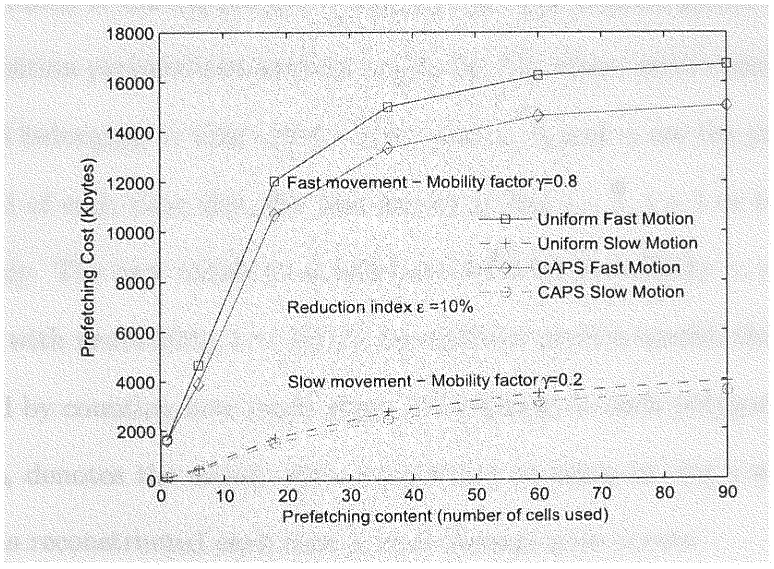


Figure 4.2: Prefetching cost (Equation 4.1 and $\epsilon = 10\%$)

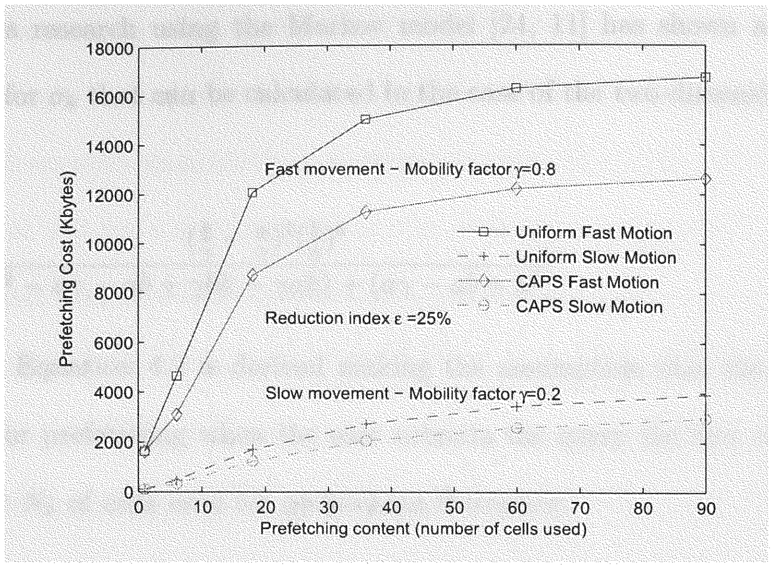


Figure 4.3: Prefetching cost (Equation 4.1and $\epsilon = 25\%$)

Next, the transition and steady state probabilities are evaluated. 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 [26, 24, 11], where state i means that the user is in a cell belonging to ring i ($0 \leq i \leq k$), 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 [24, 11] has shown an approximate expression for σ_k that can be calculated in the case of the two-dimensional motion as follows:

$$\sigma_k = \frac{(\frac{b}{\gamma} - \frac{a}{\gamma})^2(\frac{b}{a})^k}{(b^2 - a\gamma - ab + \gamma bk - \gamma ak) + (a\gamma - ab + a^2)}$$

Finally, Equation 4.2 is derived making the assumption that the number N_0 of cells used for prefetching when the user submits the query the first time is equal to the number N_k of cells used for prefetching thereafter:

$$W = (W_q + W_a N_k)(1 + t_q b_k \sigma_k) \tag{4.2}$$

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.

4.3.1 Numerical Results

In this subsection, the numerical results presented have been obtained by using the following values: $t_{loc} = 10$ msec and $t_{rem} = 5$sec. An average query duration of $t_q = 500$ discrete time units is used. A $W_a = 10$ Kb is used as the typical size of query results. Figures 4.2 and 4.3, demonstrate that using clusters with radius $k = 1$ (6 cells) for prefetching, a latency improvement is experienced 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 becomes substantially 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 $(0, 1, 2, 3, ..., k)$ 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 presented method, while maintaining the prefetching scope 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 presented modified motion 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 $P_i(k)$, where $P_i(k)$ is the prime list (Section 3.4.2) 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 as the number of cells that cover 75% of the neighboring cells (e.g. the number of restaurants in a metropolitan location). To quantify this prefetching cost savings, let ϵ be the percentage of cells reduction index, which is specific to the data item scope distribution, in this case ϵ=25%. Then, using Equation 4.2 the prefetching cost savings can be estimated. Figure 4.3 shows that for ϵ=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 mobility case and from 0% to 40% for the high mobility case.

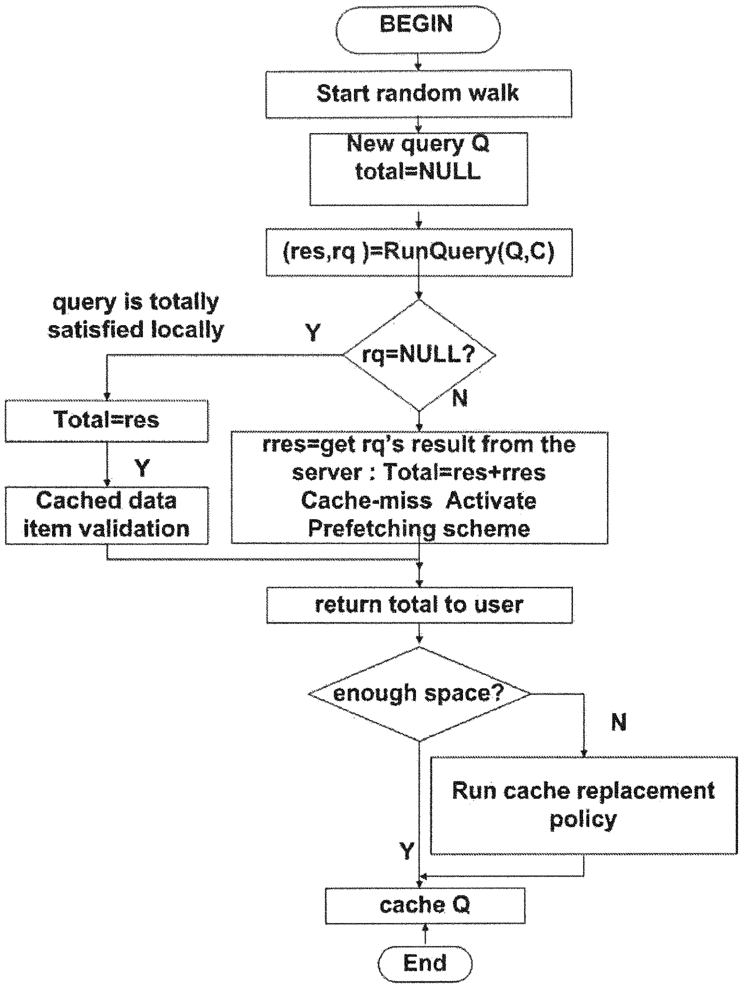


Figure 4.4: Simulation work flow (res:resident query, rq:remote query).

4.4 Prefetching Performance Evaluation

In this section, the presented location and context aware cache management strategy (CAPS) is evaluated and compared to the standard directional scheme based on tangent velocity [28] (denoted as non-CAPS 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.

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), the results for CAPS and DR are averaged. The DR scheme uses FAR [7] (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. Figure 4.4 depicts an overall data caching system model used for this simulation. Since it is well-known that prefetching will benefit from a large cache size a small cache size scale from 1% to 10% is used. However, steadily falling costs of storage lead to caches of larger sizes [48], so additional experiments are included using larger cache size levels i.e., from 10% to 50%. In the next sub-sections, the performance of the examined schemes is examined focusing on the characteristics that could impact the prefetching decisions, such as the cache size, the simulation time, the query delay, the data item types, the affinity factor, the cells numbers and the movement speed.

Finally it is noted that the presented prefetching strategy fails for uncorrelated query patterns as expected. In such extreme cases prefetching has 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, the third factor, i.e., the user profile, can be included (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).

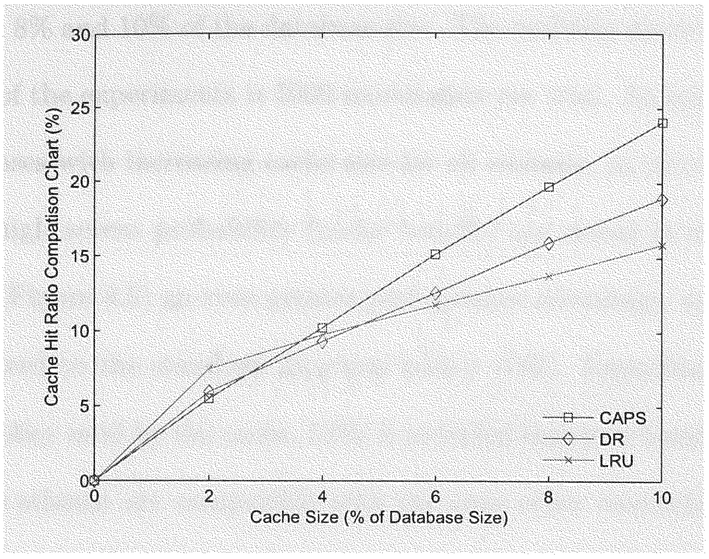


Figure 4.5: Effect of cache size in LRU, DR, and CAPS.

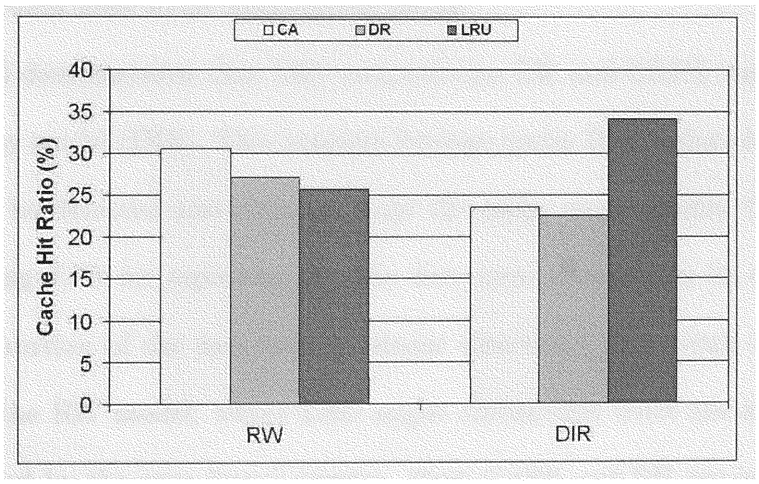


Figure 4.6: Effect of cache size for RW and DIR movements.

4.4.1 Impact of Cache Size

In this set of experiments the cache hit ratio is measured under five cache size settings: 2% , 4% , 6% , 8% and 10% of the database size. The mobility model used is the 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. CAPS demonstrates (Figure 4.5) an even greater performance advantage, approximately 0% to 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 from 5% to 15%. The performance advantage of CAPS is a result of CAPS being able to adapt itself to the data items distribution in the neighboring cells, due mainly to its data item scope distribution factor.

Figure 4.6 demonstrates that LRU outperforms DR and CAPS under the directional mobility model (DIR). This happens because under 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 CAPS and DR are independent of the recent access history, and therefore perform much better in this case.

In summary, CAPS has a more stable performance 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 Figure 4.6 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 significant caching benefit in this scenario.

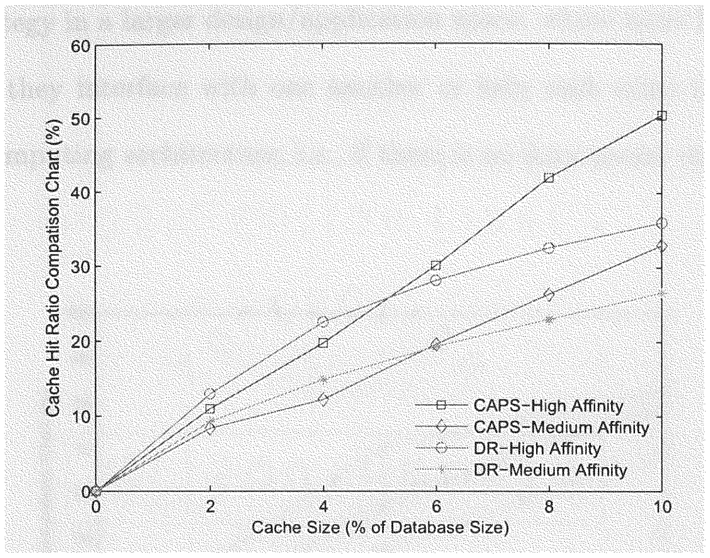


Figure 4.7: Effect of query pattern (Affinity factor) for DR and CAPS.

4.4.2 Impact of Affinity Factor

In this set of experiments, the effect of the query pattern is simulated by changing the affinity factor. The affinity factor (i.e., from $f = Low$, to $f = Medium$ and to $f = High$) determines how close the prefetching query matches the two criteria (ϕ_1, ϕ_2) defined in section 4.2.1. 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. Figure 4.7 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 CAPS can reach very high cache hit ratios as the cache size increases. This characteristic suggests the possible capabilities of the presented strategy in a larger design/application space, where more MTs can be used to examine if they interface with one another or help each other in a peer-to-peer cooperative computing architecture, i.e., if there is no data access via a BS or a MSS ask your peer.

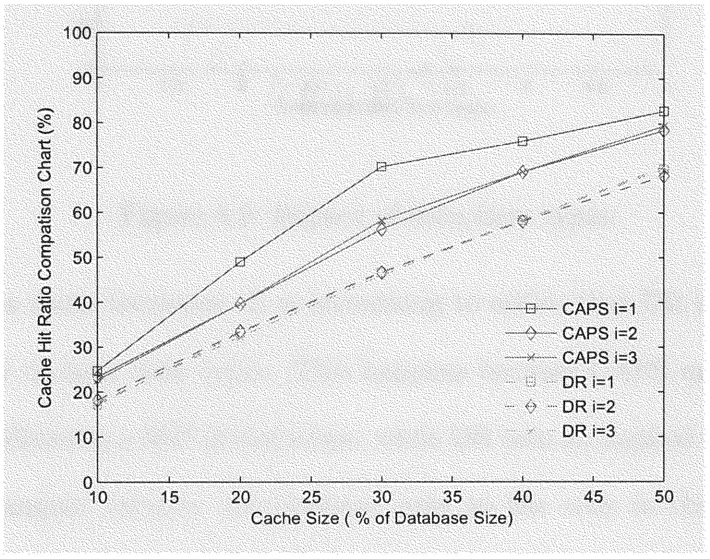


Figure 4.8: Impact of data item types - Comparison chart.

4.4.3 Impact of Data Item Types

In this set of experiments, the scalability of the presented cache management strategy is evaluated by varying the data item types from one to five and by comparing them to DR. Figures 4.8 and 4.9 show the performance results when the number of data item types is changed from "1" to "5". The performance of CAPS shows on average a 10% to 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

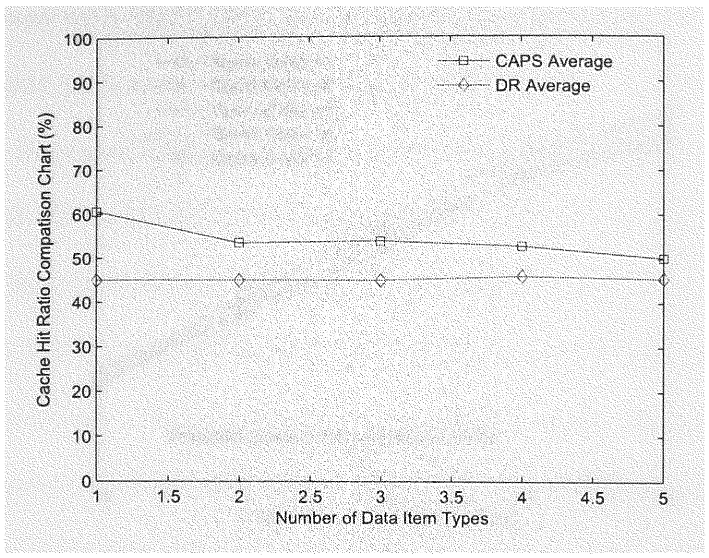


Figure 4.9: Impact of data item types.

number of data items increases. It is interesting to notice that DR is hardly affected by the number of data item types. This happens because CAPS explores its future location capabilities in a 360^0 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 take into account likely sharp turns (really random movements) of the MT. In addition, DR demonstrates a significantly higher overhead of continuous velocity calculations every Δt. Nevertheless, CAPS requires an insignificant overhead to identify the mobile user's neighboring cells and to estimate the replacement value of the cached segments. The conclusion drawn from these results is that CAPS is more appropriate for a LBS system where the database is partitioned to different volumes per data item types for higher performance.

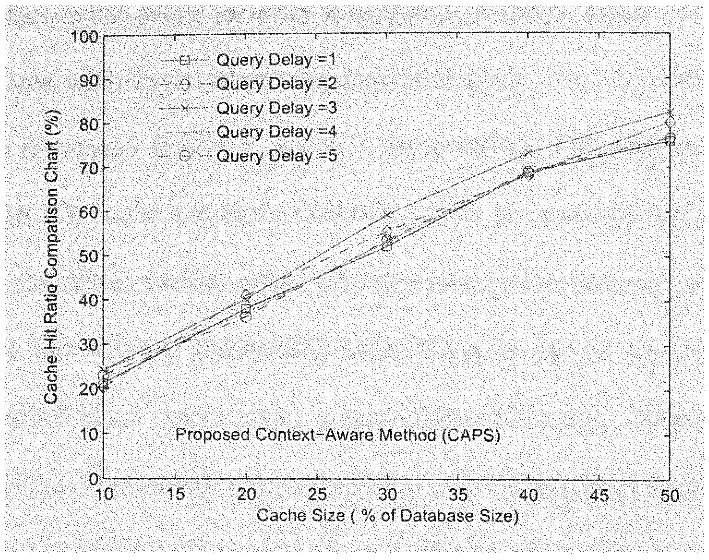


Figure 4.10: Impact of query delay on CAPS.

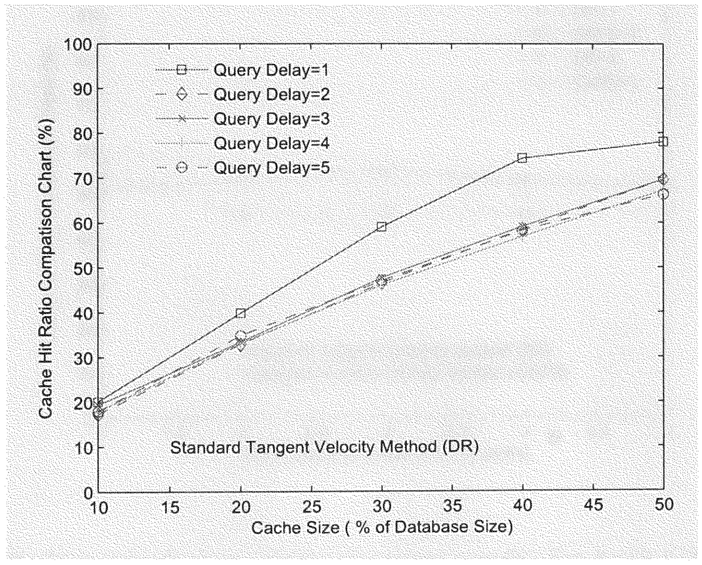


Figure 4.11: Impact of query delay on DR.

4.4.4 Impact of Query Delay

The query delay is the time interval between two consecutive client queries. In this set of experiments the impact of the query delay on the cache hit ratio is evaluated under the 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 presented 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".

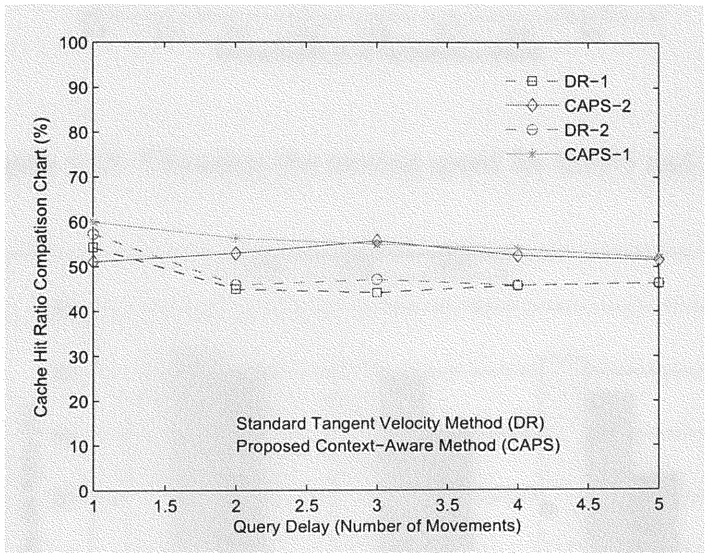


Figure 4.12: Impact of query delay-Average results for both CAPS and DR.

DR blindly prefetches information content based on the user's direction that needs to keep evaluating every Δt, hence it cannot prefetch far enough. However, CAPS can do more selective prefetching which allows it to prefetch at the longest distance. To further validate these results the experiment has been run for the regular database size (CAPS-1, DR-1) and then for a double database size (CAPS-2, DR-2) with approximately the same results (Figure 4.12). This experiment allowed us to quantify the significant advantage of CAPS in a real life random query delay scenario over the standard DR scheme.

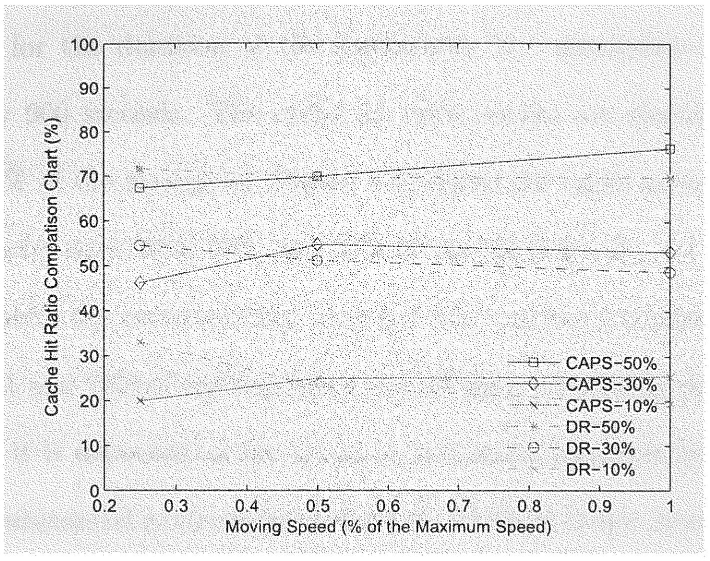


Figure 4.13: Changing the moving speed for CAPS and DR.

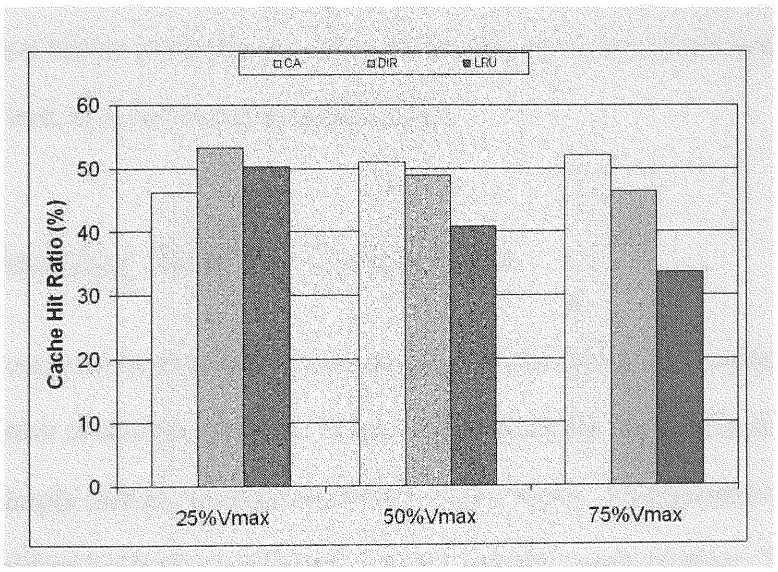


Figure 4.14: Average speed performance comparisons for CAPS, DR and LRU.

4.4.5 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 seconds. The cache hit ratio results are plotted against a scale from $10 - 100\%$ of the maxspeed. Figure 4.13 shows the cache average response time against the cache sizes 10%, 30% and 50% of the database size for CAPS and DIR. Figure 4.14 shows the 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, CAPS 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. CAPS demonstrates a much less dramatic performance reduction for higher speeds which is a significant advantage over the other policies, whereas DR demonstrates a better performance at lower speeds. As it was noted before, DR does not perform well in a fast moving environment.

4.5 Prefetching Strategy Conclusions

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. The presented prefetching strategy considers 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), an efficient future cell prediction filtering mechanism was presented based on the valid scope concept. Next, the prediction level is improved into the most likely future query pattern based on the query affinity, while the cache content with the highest benefit is preserved.

Both the performance and simulation models are based on general "context" and "user behavior" models. From the above performance results the conclusion drawn is 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. The presented context-aware future location prediction approach was used to mitigate this issue based on the data items valid scope and query content control.

Chapter 5

THE EXPERIMENTAL ENVIRONMENT

Despite the recent surge in research activities in the mobile and wireless infrastructure software simulation remains the primary approach to evaluate the data caching performance, as it is fairly easy to implement and manipulate. Data caching in a real world mobile user system is too complex to allow for an analytical evaluation through a realistic mobility model. This chapter presents the design of a random walk simulator that captures the movement of mobile users in Personal Communications Services (PCS) networks. A novel mathematical model is used and the validity of the data fetched from the server to identify a reduced subset of neighboring cells used to simulate the data caching computing paradigm for mobile computing. We demonstrate the capabilities of the simulator by simulating three key cache management replacement policies and measure the cache hit ratio performance.

A wireless infrastructure based network is simulated, where the simulated area in which a mobile user's movement takes places, is described using a NxN rectangular area of cells, represented by fixed size hexagons (other shapes may also be used). The same cell array was adopted in previous works [25], [26] and [36]. To set the simulation environment, two approaches were considered; to use the network simulator (ns-2)[49] and to develop a custom simulation environment. After analyzing the ns-2 environment, it was found that to exploit ns-2 in this dissertation, an enormous amount of changes should be applied to its internal classes. Therefore it was decided to implement a simulation environment using C++. The presented simulation method includes a heuristic mathematical model which provides a number of supplementary functions:

- Identifies the neighboring cells based on the user's current wireless cell.
- Efficiently measures the distance between any two cells.
- Uses the valid scope distribution to reduce the number of future locations.

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 presented 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 movement patterns 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 the moving direction is roughly reserved. This characteristic if combined with the mobile user's profile obviously is a better model for movements purpose.

5.1 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 ten) selected randomly. A query delay is implemented between consecutive queries. The queries used for the simulation experiments 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, which is a percentage of the database size. In order to be fair to different caching schemes, the cache contains both the space needed for storing item parameters (e.g., the scope information for FLA) and the space available for storing data. To answer a query, the client first checks its local cache. If the data value for the requested item with respect to 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. To simulate the affinity towards a selected query properly, the Matsumoto and Nishimura's random number generator 'Mersenne Twister' [46] is used, which generates numbers in a much larger range.

The affinity effect is created by the query generator using a calculated query pool (Algorithm 5). The query pool defines how large the random number needs to be. It is simply the itemTypeCount raised to the power of the queryAffinity value. Raising each item type number to the power of the queryAffinity value would end up with extremely large range between values for each of the itemTypeCount values, thus creating larger and larger pools of numbers that will result in selecting a particular queryType. Next, a random number is selected in the range 1 to queryPool inclusively. The random number is raised to the power of 1 over the queryAffinity value. This brings us back close to the original queryItem type. Next the result needs to be normalized so that it is exactly one of our queryItem values. This is achieved by taking the next highest integer, i.e., 9.2 becomes 10, 4.8 becomes 5, 6 becomes 6, etc... That value needs to be subtracted from the itemTypeCount so that the affinity is reversed. Otherwise, an affinity towards the higher values will be Achieved. An affinity towards the lower number is preferred. The movement process is controlled by a number of modules i.e., Algorithm 6 which uses a random process to select the next direction (cell). Other modules define the valid cells for each random walk.

The presented replacement algorithm implies some data item type differentiation. A number of semantic caches are simulated by the cache management process instead of having one single cache. This process removes and adds segments based on the underlined replacement policy (LRU, FAR and FLA). Algorithm 4 shows the function for creating five (10-50%) cache objects for each replacement schemes. Each object represents a semantic cache of size equal to the $10 - 50\%$ of the database size. Algorithm 6 provides the pseudo-code for the function which returns the list of cell_id's of (only) those cells used by the random walk.

Algorithm 4 Cache objects for replacement schemes

```
/* Create an array of caches 5 of each type policy scheme

Cache sizes vary from 10% to 50% of total DB size. */

for int i = 0; i < 5; i++ do
    int cacheSize = (int)(((float)(i+1)/10) * dbSize);
    cacheArray[i] = new LRUCache(cacheSize);
    cacheArray[i+5] = new FARCache(cacheSize);
    cacheArray[i+10] = new FLACache(cacheSize);
    cacheArray[i+15] = new PAIDCache(cacheSize);
end for
End
```

Algorithm 5 Query generator -Affinity effect

```
queryPool = (int)pow(itemTypeCount, queryAffinity);
int rqt = randGen → IRandom(1, queryPool);
float aqt = powf((float)rqt, (1/(float)queryAffinity));
int qt = (itemTypeCount - ((int)ceil(aqt)));
```

5.2 Cache Management Simulation Examples

In this section, the simulation modules are shown for the industry standard replacement policy LRU, the semantic caching policy FAR [7] and FLA policy [42].

5.2.1 The Physical Cache Organization

The cache is arranged to hold the most recently accessed segments of main memory. A segment of memory is referred to as a line which contains a consecutive sequence of bytes in the main memory. Typically, line lengths of 32,64 or 128 bytes are used.

Algorithm 6 simWorld::step() function

/* Giving a random direction ($n = 0, 1, 2, 3, 4, 5$)

corresponding to the directional vector

(N, NE, SE, S, SW, NW) the function will produce

the next cell_id number. */

```
1: INPUT:(direction)
2: OUTPUT:current_cell
3: bool isValidStep = false;
4: while ! isValidStep do
5:    direction = stepValue(rand() % 6);
6:    if validStep(currentCell + direction) then
7:       previousCell = currentCell;
8:       currentCell += direction;
9:       isValidStep = true;
10:   end if
11:   return currentCell;
12: end while
13: End
```

For example, Table 5.1 shows the 16 megabytes of main memory partitioned into lines that are 64 bytes long.

In 16 megabytes of real memory there are 262144 lines of 64 bytes, that is, (16 megabytes) / (64 bytes per line) = 262144 lines. A cache that has 512 lines with 64 bytes per line holds 32K bytes. As memory requests are presented to the cache, cache hardware compares the line address of the request with the line address stored in the

Algorithm 7 World DB dynamic population module

```
1: void FLAWorld::populateWorld()
2: {
3: int cellNumber, itemType;
4: for int i = 0; i ¡ _dbSize; i++ do
5:     {
6:     while !validStep(cellNumber (rand() % 225) + 1)); do
7:         itemType = rand() % _itemTypeCount;
8:         cells[cellNumber] → AddItem(i, itemType);
9:         scope[itemType]++;
10:    end while
11: end for
12: End
```

Table 5.1: Memory and Cache Line Addresses

S_0	bytes 0 through 63 are in line number 0
S_1	bytes 64 through 127 are in line number 1
S_2	bytes 128 through 191 are in line number 2
–	________________________________
–	________________________________
–	________________________________
–	________________________________
S_{262141}	bytes 16777023 through 16777087 are in line number 262141
S_{262142}	bytes 16777088 through 16777151 are in line number 262142
S_{262143}	bytes 16777152 through 16777215 are in line number 262143

Algorithm 8 FAR - Determine the direction priority module

```
int FARCache::directionPriority
(int cell, int prevCell, int target) {
int q1 = quadrant(cell, prevCell);
int q2 = quadrant(cell, target);
if $q1 == q2$ then
    return 2;
    if $abs(q1 - q2) == 1$ then
        return 1;
    end if
    return 0;
end if
}
End
```

cache directory. If the line address is found in the cache directory, a cache *hit* occurs. If the address is not found, a *miss* occurs, and the request must be satisfied from the main memory.

5.2.2 LRU Baseline Policy Simulation

The LRU replacement algorithm that is commonly used for cache designs is presented as the baseline policy. Every time a *cache miss* occurs, the replacement algorithm is invoked, and a line is discarded from the cache. The ultimate goal of any replacement algorithm is to minimize the total number of misses. The strategy used by LRU is to keep those lines that were most recently referenced and to discard the line that was referenced the furthest in the past. To do this, each directory entry has an associated age tag that identifies the chronological order of its last reference. The LRU algorithm uses the age tags to identify the line that was referenced the furthest time in the past. Once identified, this line is then discarded. Due to hardware constraints, LRU is usually restricted to discarding a line that was just referenced. Thus the age tags represent a chronological ordering relative to the lines within a conformity class. When a miss occurs, the line with the oldest age tag is discarded. A *push-down stack* is used as an abstract representation to model the replacement decisions made by an LRU algorithm (Algorithm 9). The stack represents one line in a cache and the stack depth corresponds to the set associativity used by the cache.

Let S_i, represent a stack of elements at time t. Let the elements of S be given by $S_i(1), S_i(2), ..., S_i(n)$. Let the stack S be ordered according to an age-tag that represents the time from its last reference. At the top of the stack, $S_i(1)$, is the most-recently-used (MRU) line, and at the bottom of the stack, $S_i(n)$, is the least-recently-used line. Next, assume line λ is a stack miss at time $t+1$. Then line λ is put on the top of the stack. All lines which were formerly in the stack are pushed down one position. The last line in the stack is discarded.

$$S_{t+1}(1) \leftarrow \lambda$$

$$S_{t+1}(i+1) \leftarrow S_t(i) \quad i = 1, 2, ...n-1$$

$$S_t(n) \quad is \quad discarded$$

New requested elements(objects) are inserted at the head of the list. On a hit the object is removed from its current position and inserted at the head. Replacement takes place at the end of the list. Searching can be supported by hashing techniques.

Algorithm 9 LRU Replacement Algorithm

```
1: char* LRUCache::getCacheType()
2: { return "LRU  0" ; }
3: Cache::CacheElement* LRUCache::
4: replaceElement(FLAItem *item, int cell, int qty)
5: }
6: CacheElement *ce;
7: ce = popFromTail();
8: pushToHead(item, cell, qty);
9: return ce;
10: }
11: End
```

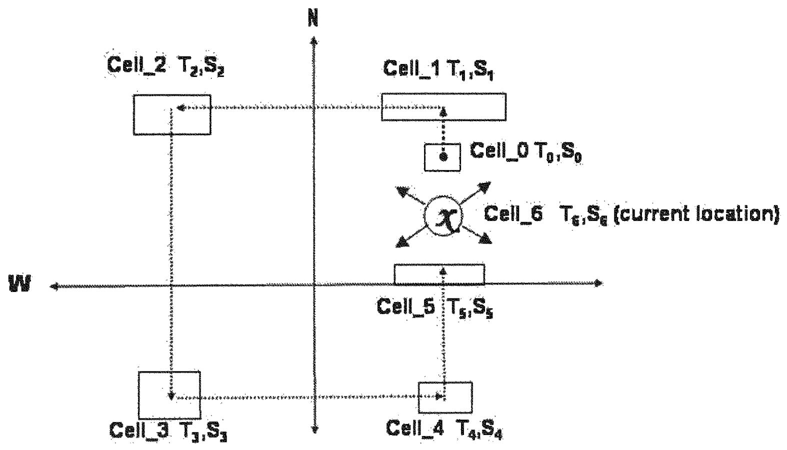


Figure 5.1: A query random walk for the FAR policy.

5.2.3 The FAR Replacement Policy

Ren and Dunham [7] developed a mobility model to represent the moving behavior of mobile users and formally defined location dependent queries. Based on their mobility model, they developed an LDD semantic cache replacement policy they name furthest away replacement (FAR). FAR chooses for replacement those segments which are not in the moving direction and are furthest from the user (see Figure 5.1). A future location is anticipated based on the current *user's direction* and *speed.* FAR presented cache replacement policy implies future location prediction based on tangent velocity.

Chapter 6

CONCLUSIONS

This dissertation makes two important contributions towards designing effective cache management architecture at the mobile client of the wireless infrastructure. The designs and formulations used can reduce both latency and network costs in mobile environments. The results obtained can form a solid foundation for future development work. This chapter summarizes the results of this research and gives several directions for future research on data caching in the mobile computing paradigm.

6.1 Contributions of this dissertation

The deployment of wireless communications coupled with the popularity of portable devices has led to significant research in the area of mobile data caching. Prior research has focused on the development of solutions that allow applications to run in wireless environment using proxy based techniques. Most of these approaches are semantic based and do not provide adequate support representing the context of the user. Even though the context may be treated implicitly it is crucial to data management.

This dissertation makes two important contributions towards designing effective cache management architecture at the mobile client of the wireless infrastructure. The designs and formulations used can reduce both latency and network costs in mobile environments. The results obtained can form a solid foundation for future development work. This chapter summarizes the results of this research and gives several directions for future research on data caching in the mobile computing paradigm.

6.2 Impact of the Results to Mobile Computing

Classical cache management strategies (i.e., LRU, MRU, FAR) are not suitable for mobile environments due to the high mobility of mobile clients. On the other hand, other presented techniques exhibit a considerable amount of complication and processing overhead. The presented mobility model is simple which makes the prediction algorithm quite efficient by avoiding complex calculations. It considers both the future prediction and the valid scope concept collaboration to efficiently implement cache replacement and prefetching due to the location changes. The presented strategies' results demonstrated the benefit of using an efficient symbolic model based on cell granularity rather than the (x, y) geometrical model used by other research teams.

The use of prefetching was analyzed in a combined approach with the cache replacement scheme. An idealized motion model was considered. However, the conclusion was drawn that for a context-aware information service, prefetching is highly beneficial for both latency and traffic reduction only when the amount of prefetched information grows linearly with the "radius" of the area covered by such information. A cost reduction approach was used to examine the effectiveness of this technique. It was determined that in some cases, prefetching is only beneficial for latency reduction, but causes a cost increase with respect to no prefetching; this issue was mitigated by the use of a reduced set of future cells derived by examining the future locations of interest to the query. In this dissertation the following contributions have been made:

- A Context-Aware Prefetching Strategy (CAPS), 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 Future Location-Aware (FLA) cache replacement policy, which uses future location-prediction and the validity of the data based on location. When a LDD query is submitted for execution, the location where the mobile client is currently visiting is directly related to the data to be retrieved. To select the cache entries for removal, a dynamically formed set of future cells called the "primary" list of cells is used. Cache entries associated with cells less likely to be visited in the future are removed first. For better efficiency, the overall replacement granularity is achieved dynamically in a zooming mode along three levels: remote ring users, closest neighboring cells and data items.

- A direct result of the proposed cache management strategies is the formation and maintenance of the *context-aware cache* of data items with a high cache benefit (value) which are included at a low cost. Data items with maximum benefit are prefetched while cache data entries with minimum benefit (highest replacement score) are evicted. The context-aware cache is then updated if the mobile user subsequently strays out of the predefined prime list of cells. Alternatively a time field may also be used for the same purpose.

- A random walk simulator is designed and implemented, that uses a novel mathematical model to provide a number of supplementary functions such as neighboring cells identifications and calculate the distance among cells.

- An intelligent algorithm was presented for the query processing at the client. The presented algorithm significantly improves this process by discriminating the cached data items based on their future location binding information, based on cell granularity.

The presented cache management strategies improve the data caching process in mobile computing, using future location prediction. The overall outcome of these techniques is the strengthening of the query's prefetching capabilities and therefore the local processing of a larger number of location dependent future queries.

6.3 Future Work

Although the presented work is extensive and forms a solid basis for incorporating new techniques into the data caching computing paradigm, much more is needed in this area of research. The following are interesting and challenging areas for future research:

1. *Network composition:* Cellular networks are not cost effective if they become fully distributed. Placing caches within cells may not always be a feasible solution to efficient data replication, since the cost of wireless bandwidth used for prefetching could dominate over the benefits of potentially better latency performance. Currently, caches are centralized to avoid huge costs in data replication. Other network types need to be investigated where the scenario

would be more credible. In addition, the cost in moving cached data around the network to place it closer to the mobile terminal would be huge compared to the penalty of accessing the data at a centralized node (as is done today) over multiple hops in the core network. In addition, the penalties involved in terms of data movement in a cellular network to maintain such a cache structure must be considered.

2. *Mobility models:* It is clear that the benefits and cost of the presented techniques are highly dependent on the ability of the cache management processes (i.e., both replacement and prefetching) to predict the mobility behavior of users. It is for this reason that a user's mobility, i.e., activity-based modeling, will require further detailed study.

   The movement of a mobile client could be modelled by a hybrid auto-switch mechanism controlled by the user's profile. Furthermore, the third factor, i.e., the user profile, can be included (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). The incorporation of the user's profile as a third factor for the cache management processes could provide a control facility for purpose movement (i.e., weekly, monthly or daily schedules of operation) and random movement. The directional movement model was shown in chapters 3 and 4 to favor the traditional LRU cache management method due to its simplicity and the fact that past items are clearly identifiable and certain not to be needed in the future. Other times the user's profile would automatically switch to the random movement model where the presented strategies would apply. As it was shown for the random movement model LRU might erroneously evict the items that are to be requested by the near future queries. However, both CAPS and FLA are independent of the recent access history, and therefore perform much better in this case.

3. *Query pattern:* Regarding the future direction of this research, a more rigorous query pattern prediction incorporating the user profile needs to be considered. The models of user query pattern are very critical to the success of replacement and prefetching since they have a direct relationship with the query prediction. As a first approach the "affinity factor" was used. However, more rigorous work is needed with the modeling of querying behavior and prediction.

4. *Cost of wireless bandwidth consumption:* The goal of the presented prefetching strategy is to pro-actively load information from a remote server (normally far from the current user's location) to neighbor cells (prefetching zone) so that the mobile client can access this information more quickly when needed. In most cases, the remote server and the cells (i.e. base stations) are connected through wireline connection. The wireless link is used only for the last hop between the mobile client and base station. However, the prefetching happens between the remote server and base stations i.e. through wireline links. So the cost here is the wireline bandwidth consumption. This is an important point since an increase of wireline bandwidth consumption can be tolerated (especially if high-speed links are available) in order to improve query performance. A simplistic cost/benefit analysis was given and a more detailed cost model needs to be presented in the future work.

5. *Cache management processes collaboration:* The previous data caching research treats cache replacement and cache coherence as separate topics. The presented

data caching overall strategy has taken a modest first step towards the collaboration of these two cache management functions by incorporating the same factors into both cache management processes.

As it was noted in the introduction by integrating data caching replacement and prefetching, these two techniques can complement each other since data caching technique exploits the temporal locality whereas prefetching technique utilizes the spatial locality of the mobile objects. However, without circumspect design the integration of these two techniques might cause significant performance degradation to each other. For instance, to provide the cache with rich information, a remote server may deliberately send all possible prefetching hints with various levels of confidences to the cache or intermediate MSS. Without any control, a MSS will prefetch every implied object into its cache, despite that the confidences of some prefetching rules may be low. In this case, a significant portion of the cache content will be replaced because a MSS may concurrently serve a large amount of client requests and each of these requests may trigger certain prefetching rules. As a result, the state of the cache content will become unstable and the cache hit ratio will drop sharply. On the contrary, if the prefetching control is over strict, a MSS will tend to discard some beneficial hints provided by the remote server, thus whittling down the advantage of prefetching.

In view of these obeservation, the motivation for future study is to design an innovative cache replacement algorithm, which not only considers the caching effect in the mobile environment but also evaluates the prefetching rules provided by various prefetching schemes.

LIST OF REFERENCES

[1] S. Acharya, B.R. Badrinath, T. Imielinski, and J.C. Navas, "A WWW-Based Location-Dependent Infromation Service for Mobile Clients," *In the 4th Int'l WWW Conference*, January 1994.

[2] C. Tait, H. Lei, S. Acharya and H. Chang, "Intelligent File Hoarding for Mobile Computers," *In Proceedings of Mobile Computing and Networking Int'l Conference*, Berkely, CA, pp. 119-125, 1995.

[3] D. Barbara and T. Imielinski, "Sleepers and Workaholics: Caching Strategies for Mobile Environmnets," *In Proceedings ACM SIGMOD Conference Management of Data*, pp. 1-12, May 1994.

[4] G. Kuenning, "Design of the SEER Predictive Caching Scheme," *In Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA., US. 1994.

[5] S. Dar, M.J. Franklin, B.T. Jonsson, D. Srivastava, and M. Tan, "Semantic Data Caching and Replacement," *In 22nd Int'l Conf. Very Large Data Base* (VLDB 96), Morgan Kaufmann, San Fransisco, pp. 330-341, 1996.

[6] M.H. Duhham, V. Kumar, "Location Dependent Data and its management in Mobile Computing," *In 9th Int'l Workshop on Database and Expert Systems Applications, DEXA 98*, pp. 414-419, Vienna, Austria, August 1998.

[7] Q. Ren, M.H. Dunham, "Using Semantic Caching to Manage Location Dependent Data in Mobile Computing," *In 6th Annual Int'l Conference on Mobile Computing and Networking* (In Proceedings of the MobiCom 2000, New York), ACM Press, pp. 210-221, August 6-11, 2000.

[8] Q. Ren, M.H. Dunham, V. Kumar, "Semantic Caching and Query Processing," *In IEEE Transaction Knowledge and Data Engineering*, vol. 15, no. 1, pp. 192-210, 2003.

[9] Q.L. Hu , D.L. Lee, and W-C. Lee, "Data Delivery techniques in Asymettric Communication Environments," *In Proceedings of MobiComm'99*, August 1999.

[10] L.D. Fife and L. Gruenwald, "Research Issues for Data Communications in Mobile Ad-Hoc Network database Systems," *SIGMOD Record, 32(2)*, June 2003

[11] V.N. Persone, V.Grassi, A. Morlupi, "Modeling and Evaluation of Prefetching Policies for Context-Aware Information Services," *In Proceedings of the 4th Conference on Mobile Computing and Networking*, pp. 55-65 1998.

[12] B. Zheng and D.L. Lee, "Semantic Caching in Location-Dependent Query Processing," *In Proceedings of the 7th Int'l. symposium on Spatal and Temporal Databases (SSTD 01)*, pp. 97-116, July 2001.

[13] B. Zheng, and D.L. Lee, "Cache Invalidation and Replacement Strategies for Location Dependent Data in Mobile Environments," *In IEEE Transactions on Computers*, vol. 51, no. 10, pp. 1141-1153, October 2002.

[14] B. Y. L. Chan, A. Si, and H. V. Leong, "Cache Management for Mobile Databases: Design and Evaluation," *In Proceedings of the 14th Intl. Conference on Data Engineering*, pp. 54-63, February 1998.

[15] S. Acharya, "Broadcast Disks: Dissemination-Based Data Management for Asymmetric Communication Environments," *PhD dissertation*, Brown Univ., May 1998.

[16] L. Tassiulas and C.J. Su, "Optimal Memory Management Strategies for a Broadcast Data Delivery System," *IEEE J. Selected Areas in Communications*, vol. 15, no. 7 pp. 1226-1238, September 1997.

[17] P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and querying moving objects," *In Proceedings of the ICDE*, Bermingham, U.K., pp.422-432, April 1997.

[18] K.C.K. Lee, H.V. Leong and A. Si, "Semantic Query Caching in a mobile environment," *Mobile computing and Communications Review*, vol. 3, no. 2, pp. 28-36, April 1999.

[19] D.L. Lee, W.C. Lee, J.Xu, and B.Zheng. "Data Management in Location-Dependent Information Services," *IEEE Pervasive Computing*, vol. 1, no. 3, July/September 2002.

[20] J. Xu, Q. Hu, D.L. Lee and W.C. Lee, "SAIU:An Efficient Cache Replacement Policy for Wireless On-Demand Broadcasts," *In Proceedings of the 9th ACM Int'l Conf. Information and Knowledge Management*, pp. 46-53, Nov. 2000.

[21] J. Xu, X. Tang and D.L. Lee, "Performance Analysis of Location-Dependent Cache invalidation Scheme for mobile Environments," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 15, no. 2, March/April 2003.

[22] J. Xu, J. Liu, B. Li and X. Jia, "Caching and Prefetching for Web Content Distribution," *IEEE Computing in Science and Engineering Magazine:*Special Issue on Web Engineering, 2004.

[23] H. Hu, J. Xu, W.S. Wong, B. Zheng, D.L. Lee and W.C. Lee, "Proactive Caching for Spatial Queries in Mobile Environments," *In Proceedings of the 21st Intl. Conference on Data Engineering (ICDE 2005)*, 1084-4627 2005.

[24] D.A. Levine, I.F. Akyildiz, and M. Naghshineh, "A Resource Estimation and Call Admission Algorithm for Wireless Multimedia Networks Using the Shadow Cluster Concept," *In IEEE/ACM Transactions on Networking*, vol. 5, No. 1. February 1997.

[25] J. S. M. Ho and I. F. Akyildiz, “Mobile User Location Update and Paging under Delay Constraints,” *ACM-Baltzer Journal of Wireless Networks*, vol. 1, no. 4, pp. 413-425, December 1995.

[26] I. F. Akyildiz, J. S. M. Ho, and Y. B. Lin, ”Movement-based location update and selective pagings for PCS networks,” *IEEE/ACM Trans. on Networking*, vol. 4, no. 4, pp. 629-638, August 1996.

[27] I. F. Akyildiz and W. Wang, ”The Predictive User Mobility Profile Framework for Wireless Multimedia Networks,” *In IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 1021-1035, December 2004.

[28] S. Park, D. Kim and G. Cho, “Improving prediction level of prefetching for location-aware mobile information service,” *In Future Generations Computer Systems, 20* , pp 197-203, 2004.

[29] J. Cai, K.L. Tan, B.C. Ooi, “On Incremental Cache Coherence Schemes in Mobile Computing Environments,” *In Proceedings of the ICDE*, pp. 114-123, 1997.

[30] L. Forlizzi, R. H. Guting, E. Nardelli, and M. Schneider. “A Data Model and Data Structures for Moving Objects Databases,” *Technical Report Informatik 260*, FernUniversitat Hagen, 1999.

[31] Z. Mao and C. Douligeris, “Two Location tracking Strategies for PCS Systems,” *In Proceedings of IEEE IC3N’99*, Boston, MA, pp. 318-323, October 11-13, 1999.

[32] Z. Mao and C. Douligeris, “Group Registration with Local Anchor for Location tracking in Mobile Networks,” *IEEE Trans. on MObile Computing*, vol. 5, no. 5, May 2006.

[33] M. Satyanarayana, “Challenges in Implementing a Context-Aware System,” *In IEEE Pervasive computing*, vol. 01, no. 3, pp.2, July-September, 2002.

[34] A. Datta, K. Dutta, H. Thomas, and D. VanderMeer, “World Wide Wait: A Study of Internet Scalability and Cache-Based Approaches to Alleviate It,” *In Management Science*, Vol. 49, No.10, pp. 1425-1444, October 2003.

[35] A. Aljadhai and T.F. Znati, “Predictive mobility support for QoS, provisioning in mobile wireless environments,” *In IEEE J. Select. Areas Communications*, vol. 19, pp. 1915-1931, October 2001.

[36] A. Hac and X. Zhou, “Locating Strategies for Personal Communication Networks: A Novel Tracking Strategy,” *IEEE JSAC*, vol. 15, no. 8, pp. 1425-1436, October 1997.

[37] M. Taylor, W. Waung, and M. Banan, “Internetwork Mobility: The CDPD Approach,” *N.J. :Prentice Hall, 1997*

[38] S. Drakatos, N. Pissinou, K. Makki and C. Douligeris, “Future Location Aware Semantic Caching In Mobile Computing,” *In Proceedings of the World Wide Congress (WWC) Conference*, San Francisco, CA, pp 569-574, May 2004.

[39] S. Drakatos, N. Pissinou, K. Makki and C. Douligeris, "A Context-Aware Prefetching Strategy for Mobile Computing Environments," *In Proceedings of the ACE International Wireless Communications & Mobile Computing (ACE-IWCMC 2006) Conference*, Vancouver, Canada July 3-6, 2006.

[40] S. Drakatos, N. Pissinou, K. Makki and C. Douligeris, "A Future Location-Aware Cache Management Mechanism for Mobile Computing Environments," *The Journal of Wireless Communications and Mobile Computing. (under revision).*

[41] S. Drakatos, N. Pissinou, K. Makki and C. Douligeris, *"A Context-Aware Cache Structure for Mobile Computing Environments," Elsevier the Journal of Systems and Software*, November 2006.

[42] S. Drakatos, N. Pissinou, K. Makki and C. Douligeris, "A Future Location-Prediction Replacement Strategy for Mobile Computing Environments," *In IEEE Wireless Communications and Networking Conference (IEEE WCNC2006)* Las Vegas, NV USA, vol. 4, pp. 2252-2260, April 2006.

[43] D. Lam, D.C. Cox, and J. Widow, "Teletraffic Modeling for Personal Communications SErvices," *In IEEE Communications Magazine*, vol. 35, no. 2. pp. 79-87, February 1997.

[44] U. Madhow, M. L. Honig, and K. Steiglitz, "Optimization of Wireless Resources for Personal Communications Mobility Tracking," *IEEE/ACM Transactions Networking*, vol. 3, no. 6, pp. 698-707, December 1995.

[45] T. Liu, P. Bahl and L. Chlamtac, "Mobility modeling, location tracking and trajectory prediction in wireless ATM networks," *In IEEE Select Areas of Communications*, vol. 16, pp. 922-936, August 1998.

[46] M. Matsumoto and T. Nishimura, "Mersenne Twister- A Random Number Generator," *In ACM transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3-30, August 1998.

[47] R.H Güting, M.H. Böhlen, C.S. Jensen, N.A. Lorentzos, M. Schneider, and M. Vazirgiannis, "A Foundation for Representing and Querying Moving Objects," *In ACM Transactions Database Systems (TODS)*, vol. 25, no. 1, pp. 1-42, March 2000.

[48] S. Podlipnig and L. Böszörmenyi, "A Survey of Web Cache Replacement Strategies," *ACM Compiting Surveys*, vol. 35, no. 4, pp. 374-398, December 2003.

[49] The Network Simulator ns-2, *http://www.isi.edu/nsnam/ns/*, June 2002.

[50] R.L. Mattson, J. Gecsei, D.R. Slutz, and I.L. Traiger, "Evaluation techniques for Storage Hierarchies," *IBM Sys. J.*, vol. 9, no. 2, pp. 78-117, 1970.

[51] P. J. Denning, "Working Sets Past and Present," *IEEE Transactions Software Engineering*, vol. SE-6, no. 1, pp. 64-84, 1980.

[52] S. Acharya, R. Alonso, M. Franklin, and S. Zdonick, "Broadcast Disks: Data Management for Asymmetirc Communications Environments," *In Proceedings ACM SIGMOD Conference Management of Data*, pp. 199-210, May 1995.

[53] P.Cao and S.Irani, "Cost-Aware WWW Proxy Caching Algorithms," *In Proceedings of USENIX Symposium on Internet technologies and Systems (USITS)*, pp. 193-206, Monterey, CA, December 1997.

[54] A. Bhattacharya and S.K. Das, "Lazi-Update and Information-Theoritic Approach to Track Mobile Users in PCS Networks," *In Proceedings of the ACM/IEEE Mobile Computing*, Aug. 1999.

[55] T. Liu, P. Bahl and L. Chlamtac, "Mobility Modeling,Location Tracking and Trajectory Prediction in Wireless ATM Networks," *IEEE J. Select Arteas Communications*, vol. 16, pp. 922-936, August 1998.

[56] E. G. Coffman Jr. and P.J. Denning, "Operating Systems Theory," *Prentice Hall, 1973.*

[57] H. M. Taylor and S. Karlin, "An Inroduction to Stochastic Modeling ," *New York: Academic 1984.*

[58] Feiertag, R.J. & E.I. Organick, "The Multics Input/Output System," *In Proceedings of the Third Symposium on Operating Principles*, pp. 35-41, Palo Alto, CA, USA 1972.

[59] M. Carey, M. J. Franklin, M. Livny, and E. Shekita, "Data Caching Tradeoffs in Client-Server DBMS Architectures," *In Proceedings of the ACM SIGMOD Conference*, pp. 357-366, May 1991.

[60] C. N. Lo and R. S. Wolff, "Estimated network database transaction volume to support wireless personal data communications applications," *In Proceedings of the IEEE intl. Conference Communications*, pp. 1257-1263, May 1993.

[61] T. Imielinski and B. Badrinath, "Mobile Wireless Computing: Challenges in Data management," *Communications of ACM*, 37(10):18-28, 1994.

[62] M. Abrams, M. Standridge, C. Abdulla, G. Williams, and C. Fox, "Caching Proxies: Limitations and Potentials", Computer Science," *Computer Science Department, Virginia Tech.* 1995.

[63] G. Liu, G. Maguire, "A Predictive Mobility Management Scheme for Supporting Wireless Mobile Computing," *Mobile Netwroks and Applications*, pp.113-121, 1996.

[64] A. M Keller and J. Basu, "A predicate-based semantic scheme for client-server database architectures," *The VLDB Journal*, vol.5 no.2, pp.35-47, April 1996.

[65] P. Scheuermann, J. Shim and R. Vingralek, "WATCHMAN: A data warehouse intelligent cache manager," *In Proceeding of the 22nd VLDB Conference.*, pp 51-62, Mumbai, India September 1996.

[66] T.M. Kroeger, D.D.E. Long and J.C. Mogul, "Exploring the Bounds of Web Latency Reduction from Caching and Prefetching," *In Proceedings of the USENIX Symposium on Internet Technology and Systems*, pp. 13-22, Monterey, California, December 1997.

[67] A. Tomkins, R. H. Patterson and G. Gibson, "Informed multi-process prefetching and caching," *In Proceedings of the 1997 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems,*, pp 100-114, ACM Press 1997.

[68] Ye, T., H. A. Jacobsen, and R. Katz, "Mobile Awareness in a Wide Area Wireless Network of Info-Stations," *In Proceedings of the 4th Intl. Conference in Mobile Computing and Networking (MobiCom 98)* , Dallas, TX, USA: 109-120.

[69] R. Ramakrishnan, "Database Management Systems," *WCB McGraw-Hill* 1998.

[70] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang, "Moving Objects Databases: Issues and solutions," *In Proceedings of the 10th Intl. Conference on Scientific and Sattistical Database Management*, pp. 111-122, July 1998.

[71] V. J. Tsotras, C. S. Jensen and R. T. Snodgrass, "An Extensible Notation for Spatiotemporal Index Queries," *ACM SIGMOD* Record, 27(1): 47-53, March 1998.

[72] G. Kolios, D. Gunopoulos, and V. J. Tsotras, "Indexing mobile objects," *In Proceedings of the PODS*, Philadelphia, PA, May 1999.

[73] G.B. Salzberg and V.J Tsotras, "A Comparison of Access methods for Time Evolving Data," *ACM Computing Surveys* Vol 31, No.2 , June 1999.

[74] C. Bettstetter, H. J. Vonel and J. Eberspacher, "GSM Phase 2+ General Packet Radio Service GPRS: Architecute, Protocols, and Air Interface," *IEEE Communications Surveys*, vol. 2 , no.3, third quarter, 1999.

[75] J. Shim, P. Scheuermann, and R. Vingralek, "Proxy cache design: Algorithmms, implementation and performance," *IEEE transactions on Knowledge and Data Engineering*, 11(4):549-562 July/August 1999.

[76] C. Aggarwal, J. L. Wolf, and P. S. Yu, "Caching on the World Wide Web," *IEEE Transactions Knowledge and Data Eng.*, vol. 11, no. 1, pp. 94107, January/February 1999.

[77] K. Cheverst, N. Vavies, K. Mitchell, and A. Friday, "Experinecs of Developing and Deploying a Context Aware Tourist Guide: The GUIDE Project," *In Proceedingss of the 6th Am. ACM/IEEE Int'l. Conference in Mobile Computing and Networking*, (MobiCom 2000), pp 20-31, August 2000.

[78] "Digital Cellular Telecommunicatinos System (Phase 2+); Location Services (LCS); Location Services Management(GSM 12.71 version 8.0.1 Release 1999)." *ETSI, TS 101 513 V8.0.1 (2000-11), November 2000.*

[79] U. Kuback, and K. Rothermel, "Exploiting Location Infromation for Infostation-Based Hoarding," *In Proceedings of the 7th Annual Int'l. Conference in Mobile Computing and Networking (MobiCom. 01)* , Rome, Italy: 15-27.

[80] S. Hadjieftymiades, V. Matthaiou and L. Merakos, "Supporting the WWW in Wireless Communications Through Mobile Agents," *Mobile Networks and Applications Kluwer Academic Publishers, Manufacturered in Netherlands*, 7, 305-313, 2002.

[81] T. Camp, J. Boleng and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communications Mobile Computing*, vol. 2, no. 5, pp. 483-502, 2002.

[82] C. Imai, N. H. Morikawa and T. Aoyama, "Prefethcing Architecture for Hot-Spotted Network," *In Proceedings of the IEEE Conference in Communications (ICC2001)*, Helsinki, Finland 2006-2010.

[83] Z. Naor and H. Levy, "Cell identification codes for tracking mobile users," *IEEE Infocom'99*, New York, NY, pp. 28-35, March 1999.

[84] R. Jain, Y.B. Lin, C. Lo, and S.Mohan, "A caching Strategy to Reduce Network Impacts of PCS," *IEEE Journal on Seloected Areas in Communications*, vol. 12, no. 8, pp. 1434-1444, October 1994.

[85] Q. Han and Nalini Venkatasubramanian, "Information Collection Services for QoS-Aware Mobile Applications," department of computer science, university of California at Irvine,2003.

[86] C. Bettstetter, G. Resta, and P. Santi, "The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks," IEEE Transactions on Mobile Computing, vol. 2, no. 3, July-September 2003.

[87] N. Adams, R. Gold, B.N. Schilit, M.M. Tsok, and R. Want, "An Infrared Network for Mobile Computers,", In Proceedings of the USENIX Symposium on Mobile and Location-Independent Computing, August, 1993.

[88] Jinbao Li, Y.L. Thai, and Jinzhong Li, "Data Caching and Query Processing in MANETs ,", Journal of Pervasive Cumpuiting and Communications, vol. 1, no. 3, September 2005.

APPENDICES

APPENDIX A. CELLS DISTRIBUTION

The distance is recorded in terms of the number of cells travelled by a mobile user at each trial. Each cell is simulated by an object structure, which in turn contains other objects which represent the data items of the database. Using the RW model, each cell can be identified using both an absolute number, which is the cell number, and a reference number as explained next.

The Cell number: Using a *15X15=225* grid, cell numbers are assigned for each cell. For each cell number "C", the row and column are given by the following formula.

$$\mathbf{R}ow(C) = int((C-1)/15) + 1 \tag{1}$$

$$\mathbf{C}ol(C) = \begin{cases} X * 2, & X < 8 \\ (X-8) * 2 + 1, & Otherwise \end{cases} \tag{2}$$

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), d is given by:

$$d = \begin{cases} |x_1 - x_2|, & if \quad |y_1 - y_2| = 0 \\ |y_1 - y_2|, & if \quad |x_1 - x_2| = 0 \\ |x_1 - x_2| + |y_1 - y_2| - D, & otherwise \end{cases} \tag{3}$$

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 equations 1 and 2 the row and column numbers for two random cell numbers 109 and 125, are (8,8) and (10,9) respectively. Next using equation 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.

The motivation for using the distance factor d is two-fold. First, a mobile client may seldom move to a cell that is far away from its shadow cluster area. Consequently, the current and neighboring cells have a much higher prefetching benefit (and a much less replacement probability). Second, even if a mobile client moves to a distant cell, it takes quite some time for it to do so. During this period of time, the data may have already been updated on the data server and, therefore, the prefetching benefit of a data item value in distant cells is useless. However if information of distance cells happens to be in memory it will have the highest replacement value.

APPENDIX B. TRANSITION PROBABILITIES EVALUATION

The movement probability, denoted by γ, can be considered as a measure of the mobile user's speed. A low price i.e., $02, 0.3$, indicates slow movement, while a value of γ=0.8 indicates fast movement. Then a_i, b_i and c_i, are the probabilities that, at the end of each time slot, the MT moves to ring $i-1$(previous ring), $i+1$(next ring) or remains in the same ring i, respectively.

A MT moves to a neighboring ring cell with probability 1/6. In Figure 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).

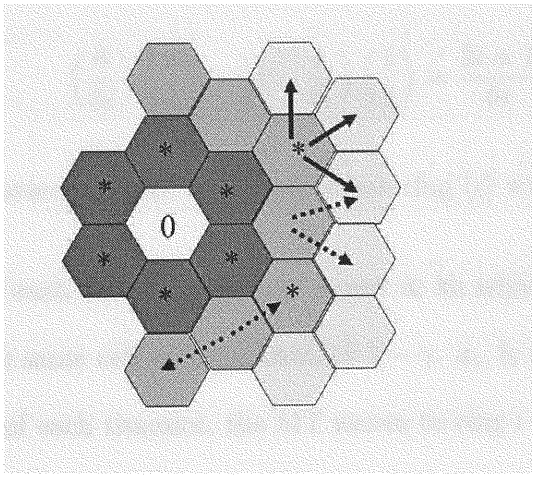


Figure 1: Layers crossing probabilities

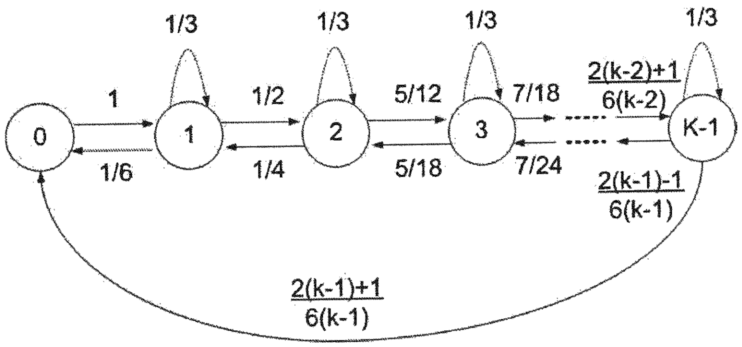


Figure 2: State diagram for the hexagonal random walk model.

If the MT is a ring cell without the * mark at ring i (see Figure 1), then it moves to ring $(i-1)$, i, or $(i+1)$ with the same probability $1/6+1/6=1/3$. The following three movement scenarios for the MT located at the ring i can take place:

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

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

2. The MT moves to a higher layer $(i+1)$ with probability:

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

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 γ, or remains in the same cell with probability $1-\gamma$. 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:

$$a_i = \begin{cases} 0, & if \quad i = 0 \\ (\frac{1}{3} - \frac{1}{6i})\gamma, & if \quad i > 0 \end{cases} \tag{4}$$

$$b_i = \begin{cases} \gamma, & if \quad i = 0 \\ (\frac{1}{3} + \frac{1}{6i})\gamma, & if \quad i > 0 \end{cases} \tag{5}$$

$$c_i = \begin{cases} 1 - \gamma, & if \quad i = 0 \\ (1 - \gamma) + \frac{1}{3}\gamma, & if \quad i > 0 \end{cases} \tag{6}$$

The state diagram derivation of the state transition probabilities given in [26, 24, 11] is shown in Figure 2, where state i means that the user is in a cell belonging to ring i, $0 \leq i \leq k$.

APPENDIX C. PERFORMANCE STATISTICAL COMPARISON

Statisticians have long used sampling techniques to draw conclusions about a large population without having to examine the entire population. They want to be able to state facts about the entire population with a high probability of being correct. Large savings in cost are possible if only a small subset of the population is needed to draw conclusions about the total population. In this section, we use the simulator's output and apply sampling techniques to construct a confidence interval (region) at a specified confidence level, to compare the hit ratio of the proposed cache management strategy (CAPS) with the industry standards semantic cache management policy FAR.

Cache hit ratios comparison: The cache management strategies CAPS and FAR were simulated for five cache sizes (10, 20, 30 ,40 and 50 % of the database) each. The sample means for the cache hit ratio are:

$\overline{x}_1 = (20.50, 44.92, 66.92, 87.19, 98.00)$, $\overline{x}_2 = (20.48, 35.13, 48.77, 62.67, 73.84)$ for CAPS and FLA respectively, compared for a time interval defined by the number of random walks. Suppose that μ_{x_1} and μ_{x_2}, are the population (i.e., all the simulation outputs) mean for each policy's cache hit ratio represented by the continuous variables x_1 and x_2 respectively. The point estimate standard deviations are s_{x_1} and s_{x_2} respectively. Let us consider the interval estimate of $d = \overline{x}_1 - \overline{x}_2$. When n is sufficiently large (i.e., n $\geq$ 30), we can establish a confidence interval for μ_d, by considering the sampling distribution of d to be approximately normally distributed with mean $\mu_d = \mu_{x_1} - \mu_{x_2}$ and standard deviation $\sigma_d = s_d/\sqrt{n}$. Writing $z_{a/2}$ for the z-value above which we find an area of $\alpha/2$, we can see from Figure 3 that:

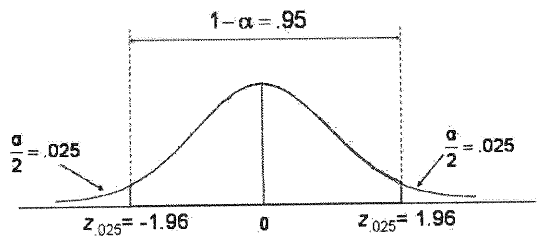


Figure 3: Random sampling confidence interval

Since in our case the sample size is small we need to use the t-distribution. A $(1-\alpha)100\%$ confidence interval for μ_d is given by:

$$\left(\overline{d} - t_{a/2,(n-1)}\frac{S_d}{\sqrt{n}}, \quad \overline{d} + t_{a/2,(n-1)}\frac{S_d}{\sqrt{n}}\right) \tag{7}$$

The values of $t_{a/2,(n-1)}$ for any confidence interval are given by (t) statistical tables found in regular statistical textbooks. Substituting the values for $\overline{d}, S_d$ in Equation 7 we get the following confidence intervals:

CAPS-LRU confidence intervals: We calculate a 90% confidence interval to be [8.11, 41.390]. Thus, because for 90% Confidence interval the true mean in a range of (8.11, 41.390) which is entirely above zero, so we can say with 90% confidence that the FLA policy is better (higher cache hit ratio).

Next, for a 95% confidence interval, we get [3.0199, 46.460]. Thus, for 95% Confidence interval the true mean in a range of (3.0199, 46.460) which is entirely above zero, so we can say with 95% confidence that the FLA policy is better (higher cache hit ratio).

CAPS-FAR confidence intervals: First for a 90% confidence interval, we get [0.5166, 21.2547]. Thus, for 90% Confidence interval the true mean in a range of (0.5166, 21.2547) which is entirely above zero, so we can say with 90% confidence that the CAPS policy is better than FAR (higher cache hit ratio).

VITA

STYLIANOS DRAKATOS

Education

09/2000-Present:	Doctoral candidate, Electrical and Computer Engineering, Florida International University, Miami Florida.
01/1976-12/1990:	Master of Science, Electrical Engineering, Ohio State University, Columbus, Ohio.
01/1974-12/1976:	Bachelor of Science, Electrical Engineering, New York Institute of Technology, New York.

Work Experience

2000-Present: Florida international University, in Miami Florida. Florida International University, College of Business Administration the department of Decision Sciences and Information Systems, lecturer.

1998-2000: OTE (A large European Telecommunications Organization). Key accounts manager, supported the key accounts management department with the design and implementation of large telecommunications and IT infrastructure projects.

1996-1999: Price Waterhouse Management Consulting Services in New York City. Principal Consultant, managed the Advanced Software Engineering Center(ASEC) in New York.

1994-1996: Bell South. Telecommunications consulting and project management, participated in the design and implementation of a number of technology integration projects.

Publications

Journals

S. Drakatos, N. Pissinou, K. Makki and C. Douligeris. *A Context-Aware Cache Structure for Mobile Computing Environments.* In the Journal of Systems and Software (Elsevier), November 2006.

S. Drakatos, N. Pissinou, K. Makki and C. Douligeris. *A Future Location-Aware Cache Management Mechanism for Mobile Computing Environments.* Journal of Wireless Communications and Mobile Computing. (under revision).

Technical Conferences

S. Drakatos, N. Pissinou, K. Makki and C. Douligeris. *Future Location Aware Semantic Caching In Mobile Computing.* In Proceedings of the World Wide Congress (WWC), San Francisco, CA, pp 569-574, May 2004.

S. Drakatos, N. Pissinou, K. Makki and C. Douligeris. *A Future Location Aware Prediction Replacement Strategy for Mobile Environments.* In Proceedings of IEEE Wireless Communications and Networking Conference (IEEE WCNC2006). Las Vegas, NV USA, Vol 4, pp. 2252-2260, April 2006.

S. Drakatos, N. Pissinou, K. Makki and C. Douligeris. *A Context-Aware Prefetching Strategy for Mobile Computing Environments.* In Proceedings of the ACE International Wireless Communications & Mobile Computing (ACE-IWCMC 2006) Conference. Vancouver, Canada July 3-6, 2006. In the ACE digital library (W3-B: Mobile Computing Symposium).