UTILITY BASED RESOURCE AWARE FRAMEWORK FOR INFORMATION CACHING AND SHARING IN MOBILE AND DISTRIBUTED SYSTEMS

by

HUAPING SHEN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2005

To my parents and Mengna.

ACKNOWLEDGEMENTS

First, I would like to thank my two supervising professors Dr. Sajal K. Das and Dr. Mohan Kumar. Both of them have given me great inspiration and excellent guidance during the course of my research work. Their support, care and patience helped me get over various hurdles during my graduate study. They have provided me with a perfect balance between guidance and freedom, which allowed me to pursue my own ideas along the right direction. I will always be grateful for their consistent encouragement.

Second, I would like to thank the members of my thesis committee, Dr. Behrooz Shirazi, Dr. Gergely Zaruba and Dr. Ramesh Yerraballi for helpful discussions and insightful comments that have greatly helped me to improve my work.

I am grateful for the support and assistance from all current and previous members of Center of Research in Wireless Mobility and Networking (CReWMaN). I would like to extend my special thanks to Zhijun Wang from whom I have learned a lot. His collaboration, insightful discussions and comments improved the quality of my work. I would also like to thank Mary Suchitha Joseph for her collaboration.

The work presented in this dissertation was supported by the Texas Advanced Research Program under Grant Number 14-771032. I am also thankful to the Department of Computer Science and the graduate school at UTA, Texas Telecommunications Engineering Consortium (TxTEC), and Nokia for financial support.

Last but not least, I would like to thank my family members for their greatest love and caring. Without their continuous support, I could not have today's achievements. I am, and will be deeply indebted to my parents during my whole lifetime. I am also

indebted to my wife Mengna who is the true companion for me in both my work and life. This dissertation is dedicated to them.

August 8, 2005

ABSTRACT

UTILITY BASED RESOURCE AWARE FRAMEWORK FOR INFORMATION CACHING AND SHARING IN MOBILE AND DISTRIBUTED SYSTEMS

Publication No. _____

HUAPING SHEN, Ph.D.

The University of Texas at Arlington, 2005

Supervising Professors: Sajal K. Das and Mohan Kumar

The Internet is evolving into an indispensable service delivery infrastructure and infinite information database. Along with the technology advancements in mobile and wireless networks, ubiquitous information service is becoming a reality in which users can access information *anytime anywhere*. However, the user mobility, network heterogeneity and resource constraints impose significant challenges to provide ubiquitous information services. In this dissertation, a utility based resource aware framework is proposed to enhance ubiquitous information availability to mobile users through data caching and peer-to-peer sharing. The framework considers the constrained resources of mobile and distributed environments and provides flexible, efficient and scalable data access services to the mobile users. The major contributions of this framework are as follows. First, we introduce a novel energy and bandwidth efficient data caching mechanism, called *GreedyDual Least Utility* (GD-LU), to enhance dynamic data availability to mobile users in cellular networks. Based on the *utility function* derived from our analytical model, we propose algorithms for cache replacement and passive prefetching of data objects.

Second, we introduce a novel scheme called *energy efficient peer-to-peer caching with optimal radius* (EPCOR), to enable peer-to-peer information sharing in multi-hop hybrid networks. In EPCOR, a peer-to-peer overlay network is built among the mobile users to facilitate cooperative sharing of data based on network proximity and data preference. In order to conserve energy, each mobile user shares a data item in a *cooperation zone*. An algorithm is developed to determine the optimal radius of the cooperation zone. Third, we investigate location-aided information retrieval in large-scale mobile peer-to-peer (MP2P) networks. A novel scheme, called *Proximity Regions for Caching in Cooperative MP2P Networks* (PReCinCt) is designed to utilize location information to support scalable data retrieval. In the PReCinCt scheme, the network topology is divided into geographical regions where each region is responsible for a set of keys representing the data. Each key is then mapped to a location based on a geographical hash function. We evaluate and validate the developed algorithms both analytically and experimentally. We have conducted extensive experiments using large scale simulations to evaluate the performance of proposed framework. Our analytical and experimental results show that the framework can efficiently provide ubiquitous information services in mobile and distributed environments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Today's Internet has become an important service delivery infrastructure rather than merely providing host connectivity and best-effort data transmission. Along with the advancements in mobile devices and wireless communication technologies, ubiquitous information access is becoming a reality, in which users can access information whenever and wherever. However, limited battery power and memory, scarce wireless bandwidth, user mobility and network heterogeneity impose significant challenges to users to access data in mobile and distributed computing environments. In this dissertation, we propose a utility based resource aware framework to enhance information availability and cooperative sharing for users in mobile and distributed environments. In this chapter, we discuss the background and challenges of our research, summarize the major contributions and organization of this dissertation.

1.1 Research Background

With the advent of ubiquitous computing era, the computer systems have been extended to the whole physical space and receded into the background of our lives [93]. Ubiquitous computing promises to provide information access services whenever and wherever. While the wireless communication infrastructure is and will continue to be characterized by a heterogeneous multitude of systems, next generation mobile systems focus on seamlessly integrating the existing wireless technologies. Such all-IP based networks will allow users to use any system anytime, anywhere. Users carrying an integrated terminal will be able to use a wide range of applications provided by multiple wireless

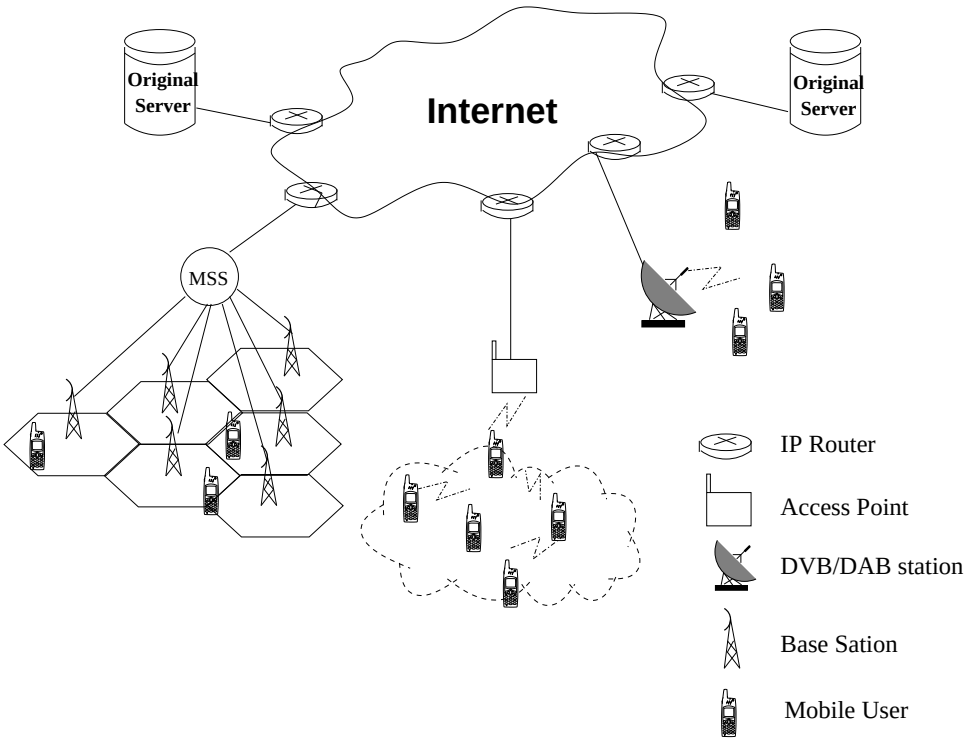

Figure 1.1 Conceptual Architecture of Next Generation Mobile Systems.

networks. Users will also have the flexibility to use multiple services from multiple service providers at the same time [43]. The communication will not only be limited to cellular telecommunication systems, like Global System for Mobile Communication (GSM), General Packet Radio Service (GPRS) or Universal Mobile Telecommunications System (UMTS), but will also include wireless LAN (WLAN), Digital Video Broadcast (DVB) [22], Digital Audio Broadcast (DAB) [21] and mobile ad hoc networks (MANET). As these networks vary in bearer service offerings, coverage, quality of service, and ability to serve a wide range of communication scenarios, the end-systems are required to provide access to all currently available wireless communication systems and adaptively use the most appropriate communication infrastructure in the right situation. Figure 1.1 shows a conceptual architecture of next generation mobile systems.

According to different wireless services of next generation mobile systems, we can classify wireless information access methods into four main categories: 1) infrastructure based approach; 2) peer-to-peer approach; 3) hybrid approach; and 4) push based approach; In the first approach, there exists a fixed information access point in the system, with which each mobile user (MU) interacts to retrieve desired information. The fixed information access point can be a base station (BS) of cellular networks or an access point (AP) of wireless LAN connected to the wired Internet, or a wireless data repository server such as an *infostation* [38]. The MUs need to send requests to the information access point to fetch the data items. We call this information access mechanism as the *infrastructure-based* approach. However, due to the expensive spectrum license fees and limited coverage area of high speed wireless networks, the deployment cost of sufficient base stations, access points or infostations to provide ubiquitous information service is prohibitive. Moreover, in some scenarios, the infrastructure may not exist, like moving vehicles, subway stations, battle field and disaster relief operations. Thus, we need another approach to provide ubiquitous information services in such scenarios.

Mobile ad hoc networks were designed for environments that lack established infrastructure. Communication networks for such scenarios as battle field and disaster relief operations are examples of ad hoc networks, where each MU retrieves the desired information from other MUs through multi-hop paths. Here MUs cooperatively act as routers to relay the data traffic, so that all MUs in the network form a mobile peer-to-peer (MP2P) system to cooperatively share their information. We call this information access scheme as the *peer-to-peer based* approach. However, due to the absence of stable network connection and frequent changes of the network topology, it is hard to provide reliable ubiquitous information services in this approach.

Recently, hybrid information access approaches [64] [59] have been proposed to complement ad hoc and infrastructure based information access methods. In the hy-

brid approach, the base stations are assumed to have Internet connection or infostation attachment. In such systems, an MU in the transmission range of a BS can retrieve information from the latter directly. When an MU moves out of the BS transmission range, it can continue to access information by relaying data to other MUs along the path to the BS. Since the hybrid approach extends the wireless service area at low cost, it is an attractive alternative for ubiquitous data services. However, due to uneven load distribution and high bottleneck at the base station, this approach suffers from scalability issues [81].

In order to provide ubiquitous information services to a large number of mobile users, digital broadcast networks, e.g., digital audio broadcast (DAB) and digital video broadcast (DVB), offer mass media services. Such networks are able to efficiently reach large groups of subscribers and provide them with a high-rate, though one way data broadcast services. In such networks, a set of information units (Web pages, digital audio or video clips, games) are periodically broadcast over the networks. Mobile users listen to the broadcast channel and acquire their requested data items. The push of information and periodic transmission cycles will enhance the information availability to mobile users. We call this approach as *push based* information access method. However, due to the nature of asymmetric communication, this approach cannot provide interactive information services.

As explained, the above four approaches have their own advantages and disadvantages to provide information services to users for different application scenarios. These four approaches are expected to coexist to provide ubiquitous information services to users in next generation mobile and distributed systems. Considering the persistence of access heterogeneity and the continued co-existence of systems, a clear requirement for next generation systems [35] is to cope with multiple air interfaces rather than to

unify and replace them. Future systems have to combine all these information access approaches to provide a seamlessly ubiquitous information service.

1.2 Research Challenges and Motivation

To enable ubiquitous access to information services for mobile and wireless users, we need to propose a middleware framework that is independent of air interface and communication services. This framework resides within the user end terminals. Before introducing the framework we first describe the specific research issues and challenges our research has been focused on. To the best of our knowledge, these challenging problems have not been solved by existing related work that will be reviewed in Chapter 2.

- **Efficient Resource Utilization.** In mobile and distributed systems, mobile devices have limited battery power, computation capacity and memory storage. Especially, as more and more bandwidth is available in wireless networks, transmitting and receiving data cost more energy to mobile users. With slow progress in the battery technology, energy of mobile devices is the crucial resource in mobile and distributed systems. In cellular networks, the wireless bandwidth is also a limited resource. Moreover, for mobile peer-to-peer communication, it is a challenge to efficiently distribute service provisioning load among different mobile users to achieve optimal load sharing as well as resource utilization.
- **Consistent Information Availability.** In mobile distributed environments, mobile users connect/disconnect from wireless networks frequently, and they are always roaming between different wireless networks. On the other hand, information is subject to dynamic changes all the time. Therefore, developing mechanisms to provide consistent information to mobile users is also the significant challenge.
- **Flexibility.** The third challenge is to achieve flexible and seamless information services to mobile users. Most existing solutions can only support information

services in specific wireless networks. As a result, the flexibility and efficiency of existing approaches are limited by the requirement of specific underlay network infrastructure support. However, heterogenous networks will co-exist to provide information services in the future. Thus, there is a need for new flexible solutions that can integrate different mechanisms and adapt to various network environments.

- **Scalability.** The challenges of scalability are twofold: user scalability and data scalability. User scalability demands a solution that can support a massive number of mobile users. On the other hand, data scalability requires mobile users to access large volumes of different types of information, like Web pages, digital audio or video clips, electronic newspapers, software, and games.

In the mobile and distributed environments like an airport, a commercial center, a campus or an urban environment, users usually access local and general information, such as news headlines, weather reports, sports, maps, music, video files or mobile games. The information requested in such environments shows high spatial locality. According to the measurement conducted in a wireless campus network [51], more than 20% of all data object requests were from nearby users or those in close proximity within the last one hour. Therefore, cooperative *peer-to-peer sharing* of frequently accessed data objects can significantly reduce the energy and bandwidth consumption due to users' request in such environments. Moreover, *Caching* is considered as one of the important techniques to relieve bandwidth constraint imposed on mobile and distributed environments [2]. Copies of remote data can be kept in the local storage of mobile devices to substantially reduce data retrievals from the original server.

Based on these two observations, in this dissertation we develop a utility based resource aware framework for information caching and sharing to support ubiquitous information services in mobile and distributed environments. In this framework, *utility functions* are used to optimize cache management and get optimal performances for

user data accesses under different wireless networks. The utility functions consider various resource constraints (e.g., energy, bandwidth, memory, network connectivity, etc.) of mobile and distributed environments. Since different utility functions can adapt to different wireless networks, the framework provides a flexible solution for information services in different application scenarios. Moreover, efficient data consistency maintenance schemes are deployed in the framework to enhance consistent information availability to users. Since the framework efficiently integrates various information providing models such as push-based, pull-based and peer-to-peer based, it can support large scale information services in mobile and distributed environments.

1.3 Contributions of the Dissertation

In this dissertation, we propose a utility based resource aware framework to enhance consistent and dynamic information availability to users in mobile and distributed environments through data caching and peer-to-peer data sharing. The framework considers the constrained resources of mobile devices and wireless networks and provides flexible, efficient and scalable data access services to the users. In this framework, there are four major components: (1) Energy-efficient data caching in cellular wireless networks; (2) passive prefetching in digital broadcast networks; (3) Peer-to-peer data sharing in multi-hop hybrid wireless networks; and (4) Location-aided data retrieval in mobile peer-to-peer systems. Figure 1.2 describes the internal components of the proposed framework.

In the proposed framework, we provide a unified information query interface that allows users to input their queries. Users can input Universal Resource Identifier (URI) [9] of requested data item, like *www.cnn.com* or input the description of requested data item, like *hero.mp3*. In order to efficiently manage large scale data items, the query interface implements a conflict free hash function, like MD5 [74] or SHA-1 [27], to hash users' query into m-bit *identifier*. Therefore, different data items have different identifiers

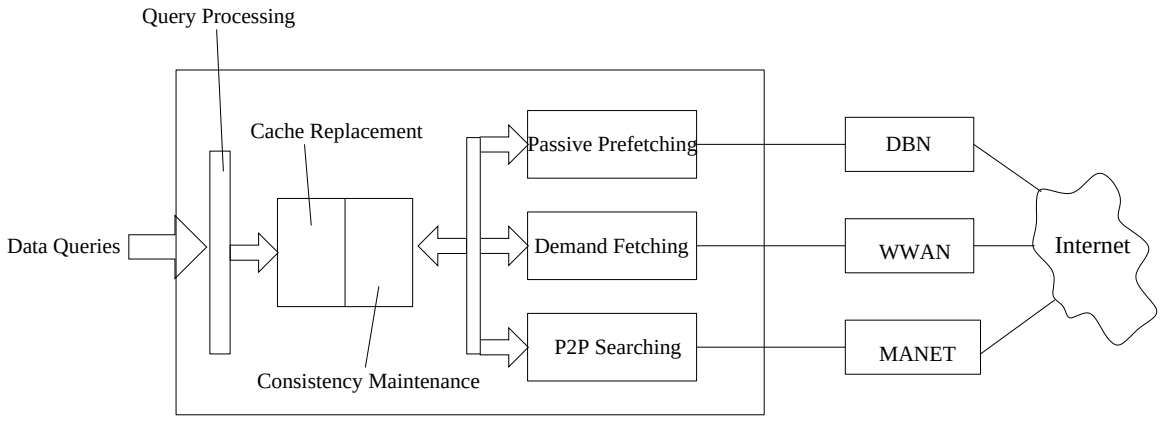

Figure 1.2 Internal Components of Proposed Framework.

(IDs). The framework uses IDs to manage all data items in cache, and uses IDs to locate and retrieve data items in mobile and distributed environments.

For each user query, the framework will search the local cache of mobile devices to check if the requested item is already cached by matching IDs of cached items and the requested item's ID. If requested item is found in the local cache, the data consistency algorithm implemented in cache management module is used to check the consistency of the item. The consistent data item is returned to the user. In case of a cache miss, the framework needs to consult with the service handoff module of mobile devices to check the availability of wireless information services. According to the availability of different information services, corresponding data retrieval schemes are used to fetch the requested item from other mobile users or from the Internet.

As mobile users access information through cellular wireless networks (e.g., GSM, GPRS, UMTS), they are subject to long access latency and high energy consumption due to the limited wireless channel bandwidth. In the framework, we propose an energy-efficient data caching scheme to reduce access latency and energy consumption of information access in the cellular networks. In the proposed scheme, we develop an analytical model for energy consumption of mobile devices due to dynamic data access. The an-

alytical model considers different events such as data request, data update, connection-disconnection and device mobility handoff and affect energy consumption due to data retrievals. Based on this model, we derive a *utility function* for each data item in terms of energy saving. A novel caching mechanism, called GreedyDual Least Utility (GD-LU) [77], is proposed for cache management in mobile devices.

In addition, mobile users access the information through digital broadcast based wireless networks (DBN), such as Digital Audio Broadcast (DAB) and Digital Video Broadcast (DVB). These systems will continue to push information (Web pages, digital audio or video clips, personal electronic newspapers) to mobile users. Under push-based information service model, we propose a passive prefetching [79] scheme to further reduce the access latency of each mobile user. In this scheme, mobile devices acquire the data items from broadcast channels for the users' future requests based on data items' *relative utility* values. A threshold value is used to admit the data items from broadcast channel. Based on the utility model, the threshold value is dynamically determined to achieve near-optimal performance tradeoff between access latency and energy consumption.

When mobile users access information service in hybrid wireless networks, our framework proposes a novel scheme called *energy efficient peer-to-peer caching with optimal radius* (EPCOR) [78], to efficiently support ubiquitous data accesses in hybrid wireless networks. In EPCOR, each mobile user or peer has a cache to store the frequently accessed data items. Cached data items in each peer node satisfy local queries as well as those from peers. Furthermore, a localized overlay network is built among mobile users to help them cooperatively share cached data based on network proximity and data preference. The neighbor relationship of two peers in the localized overlay is maintained by proactive exchange of cache index messages between them. In order to conserve energy, message exchanges are localized in a *cooperation area*, and the size of cooperation zone is determined by setting a hop-to-live value of each cache index message. We de-

velop a utility based analytical model to evaluate the performance of EPCOR system. An iterative algorithm is developed to determine the optimal radius of the cooperation zone based on the gained utility due to cooperation. Both theoretical and simulation results show that EPCOR can significantly improve the performance by saving energy, reducing access latency and balancing load of information access in hybrid wireless networks.

In order to improve the scalability performance of data retrieval in mobile ad hoc networks, we develop a location-aided data retrieval scheme in the proposed framework to support information access in environments without Internet connection. A novel scheme, called *Proximity Regions for Caching in Cooperative MP2P Networks* (PReCinCt) [80] [47] is introduced. In this scheme, the entire network topology is divided into geographical regions, each being responsible for a set of keys representing the data of interest. A hash function is used at each peer to map a key to a location in a region, called the *home region* of the key. When a peer requests a data item represented by a key, the peer obtains the location information of home region by using the hash function, and then a request message is sent to the home region by using geographic-aided routing protocol. After reaching the home region, localized flooding is used to locate the peer holding the requested data. By routing to regions rather than to specific locations, PReCinCt requires only approximate location information of each region which makes it robust against errors in location measurement and frequent mobility of peers.

After data items are retrieved, the framework uses different utility functions to evaluate the importance of retrieved items considering different application scenarios. GreedyDual (GD) based caching algorithms are deployed to manage the cache content of mobile users. In order to provide consistent information to users, various data consistency maintenance schemes are deployed in the framework to maintain the data consistency of cached data items.

To the best of our knowledge, the research work completed in this dissertation is the first comprehensive attempt to build a resource aware framework for mobile devices to support ubiquitous information access in next generation heterogenous wireless networks. Novel algorithms and functional modules have been developed and thoroughly evaluated with comprehensive analytical and simulation experiments.

1.4 Organization of the Dissertation

The rest of this dissertation is organized as follows. Chapter 2 describes research work related to this dissertation. Chapter 3 describes a utility based model for energy consumption in mobile devices. A novel caching mechanism, called GreedyDual Least Utility (GD-LU), is proposed for cache management in mobile devices. The passive prefetch algorithm is also proposed to access information in data broadcast networks. Comprehensive simulations are conducted to evaluate the performance of GD-LU cache management mechanism and passive prefetching scheme.

In Chapter 4, we extend the information sharing into multi-hop wireless networks. A mobile peer-to-peer information sharing system, EPCOR, is proposed. A model of the proposed system, and some analytical results are presented to evaluate its performance. The implementation issues and complexities of EPCOR are discussed. Simulation results verify EPCOR performances.

In Chapter 5, we investigate location-aided data retrieval in large-scale mobile peer-to-peer systems. A novel scheme, Proximity Regions for Caching in Cooperative MP2P Networks (PReCinCt) is proposed to efficiently support location-aided data retrieval in large-scale MP2P networks. Analytical performance evaluation and simulation results of PReCinCt scheme are also presented. Chapter 6 concludes the dissertation and discusses future research.

1.5 Summary

In next generation wireless networks, different wireless networks will co-exist to provide ubiquitous information services. Each mobile user will have multiple wireless interfaces so that they can operate in different wireless networks. Mobile users are deployed with service handoff to discover and select wireless systems and provide seamless mobility support under heterogenous wireless networks. With limited battery power and memory storage, a utility based framework is proposed for mobile users to support ubiquitous information services under the heterogenous wireless networks. In this section, we present an architectural overview of the proposed framework. The framework integrates three different data access schemes, i.e., passive prefethcing, on-demand fetching and peer-to-peer sharing, to make mobile users adapt to various information access model under different wireless networks. Different utility functions are proposed to manage and optimize the cache content for different data access schemes. Data consistency schemes are also deployed in the framework to provide consistent information to users. All schemes of the proposed framework are developed, analyzed and evaluated in the following chapters.

CHAPTER 2

RELATED WORK

In this chapter, we discuss the related work of this dissertation. A range of different research efforts are related to our research work. We classify the related work into several categories, each of which is briefly reviewed. Existing research work on data broadcast, caching, prefetching, cache consistency maintenance and peer-to-peer network are mainly presented in this chapter.

2.1 Data Broadcast

Data broadcasting has been considered as a promising way of disseminating and sharing information to a massive number of users in a wireless communication environment. For data broadcasting, three kinds of broadcast schemes are proposed: *push-based*, *pull-based* (or on-demand) and *hybrid*.

In push broadcast, data are periodically broadcast on the wireless channel according to the broadcast program generated by the data scheduling algorithm. The scheduling algorithm may make use of precompiled access profiles in determining the broadcast program. Once a mobile user issues query for a data item, the MH tunes into the broadcast channel and acquires the desired data, perhaps after a wait period. In the following, we describe three typical push based broadcast schemes: *flat broadcast*, *broadcast disks*, *square-root rule broadcast*.

Flat broadcast is the simplest scheme for data broadcast scheduling. With a flat broadcast program, all data items are broadcast in a round-robin manner. The access time for every data item is the same, i.e., half of the broadcast cycle. This scheme is

simple, but its performance is poor in terms of average access time when data access probabilities are skewed.

In [2], a repetitive broadcast technique, called *broadcast disk* is proposed to structure the broadcast in a way that provides improved performance for non-uniformly accessed data. The broadcast disks superimpose multiple disks spinning at different speeds on a single broadcast channel-in effect creating an arbitrarily fine-grained memory hierarchy. In this scheme, the data items are sorted according to their popularity. The list of all items are partitioned into multiple ranges (i.e., *disk*). Each of the disks is assigned a relative frequency of broadcast. Each disk is split into a number of smaller units, namely *chunks*. The broadcast program is then created by interleaving the chunks of each disk.

In [88], a *square-root rule* is discovered to achieve the minimum overall access latency for push-based data broadcast. *Square-root Rule:* Give the demand probability p_i of each item i, the minimum overall mean access time, t, is achieved when frequency f_i of each item i is proportional to square root of its demand probability $\sqrt{p_i}$ and inversely proportional to square root of its length $\sqrt{l_i}$, assuming that instances of each item are equally spaced.

In pull-based broadcast, the server provides on-demand data access in response to explicit client requests. The increased broadcast capacity due to various technological advances has spawned a number of new on-demand applications which exploit pull-based broadcast infrastructure. Once the client makes a request for a data item, it monitors the downlink channel for the server response and pulls of off the pages comprising its desired data item. The broadcast pages are assumed to have self-identifying headers and the server delivers pages comprising an item *in order*. Since the channel uses broadcast, a client's request may be satisfied by the response to a prior request for the same item from another client (in point-to-point case, this is not feasible). The amount of common requests among clients, of course depends on the application. The broadcast server queues

up the client requests as they come along unless there is already a pre-existing request for the same data item which has not yet been satisfied. The scheduling algorithm used by the server determines the order in which the requests are serviced.

Preemption [3]: Preemption involves interrupting a broadcast to service others requests before resuming the remainder of the original broadcast. According to the definition of *preemption*, the pull-based data scheduling algorithms are classified into two categories: *preemptive scheduling* and *non-preemptive scheduling*.

Pull-based broadcast scheduling for application with variable data item sizes was studied in [3]. To evaluate the performance for items of different sizes, a new performance metric called *stretch* was introduced:

Stretch: the ratio of the access time of a request to its service time, where the service time is the time needed to complete the request if it were the only job in the system.

Compared with access time, stretch is believed to be a fairer metric for items of variable sizes since it takes into account the size (included in service time) of a request item.

In the hybrid push based system, a less popular item if requested, is sent immediately after having broadcast the most popular item. Suppose D is the total number of unique data items present in the system, the push set contains data items 1 to K, that are most popular, and the remaining $(D-K)$ items are served as push set data. The hybrid broadcast architecture is first investigated in [4]. An algorithm, called *Interleaved Push and Pull* (IPP) was proposed to combine push-based and pull-based broadcast schemes. IPP mixes both push and pull by allowing clients to send pull requests for misses on the backchannel while the server supports a broadcast disk plus interleaved response to the pulls on the front channel. The allocation of bandwidth to pushed and pulled pages is determined by the $PullBW$ parameter. A refinement to IPP uses a fixed threshold to

limit the use of the backchannel by any one client. The client sends a pull request for page p only if the number of slots before p is scheduled to appear in the periodic broadcast is greater than the threshold parameter called *ThresPerc*. Threshold is expressed as a percentage of the major cycle length (i.e., the push period). When $ThresPerc = 0\%$, the client sends requests for all missed pages to the server. When $Thresperc = 100\%$ and the whole database appears in the push schedule, the client sends no request since all pages will apppear with a major cycle. Increasing the threshold has the effect of forcing clients to conserve their use of the backchannel and this, minimizes the load on the server. A client will only pull a page that would otherwise have a very high push latency. The experimental results in [4] show that when there is little or no contention at the server, pulling pages using the backchannel with no server push (*pure pull*) is the best strategy. If the server is lightly loaded, all requests get queued and are serviced much faster than the average latency of pages on the Broadcast Disk. When the server is saturated, the best strategy is to use *Pure-Push* since it provides a "safety net" and puts an upper bound on the delay for any page.

2.2 Data Consistency Maintenance

In the literature, there are three types of cache consistency maintenance algorithms for wireless mobile computing environments: stateless, stateful and hybrid. In the *stateless* approach [6], the server is unaware of client's cache content. The client needs to check the validity of cached entries from the server before each query. Even though stateless approaches employ simple database management, their scalability and ability to support disconnectedness are poor. On the other hand, *stateful* approaches [32] are scalable, but incur significant overhead due to server database management. In the hybrid approach [91], the server keeps very little information about client cache statutes. Light computa-

tion overhead is need at server database management, therefore it achieves good tradeoff between scalability and efficiency.

In the *stateless* approach, an MSS assumes no knowledge of MU's cache contents. An MSS simply sends invalidation reports (IRs) to its MUs periodically. At an MU, a data object request cannot be serviced until the next IR from the MSS is received. Synchronous methods of cache invalidation are based on periodic broadcasting of IRs. A client has to listen to the invalidation report first in order to conclude whether its cache is valid or not. (Notice that this adds some latency to query processing.) If the decision is negative the MU has to issue a query to the server and refresh its cache. It may turn out that the invalidation report leads to a *false alarm*, and in fact the cache was valid. However if given an invalidation report, the MU concludes that the cache is valid, it must in fact be so. Hence, this scheme will only allow false alarm errors and will always correctly inform the MU if his copy is invalid.

The server timestamps each report with the time at the initiation of the broadcast. If the last report was broadcast with timestamp T_i and a MU determines that a particular item's cache is valid after listening to the report, this cache gets timestamped with the value T_i (marked as valid up to this time). If the MU has to submit an uplink request because the cache is invalid, then the obtained copy has the timestamp equal to the timestamp of the request. The broadcast of the invalidation reports divides the time into intervals. Notice that the MU has to wait for the next invalidation report before answering a query. The MU keeps a list of items queried during an interval and answers them after receiving the next report.

The stateless server sends the invalidation reports according to the different strategies. We can classify the stateless approach into two categories as follows:

- Asynchronous: Here invalidation reports are broadcast immediately after changes to data items occur. In particular, the report may just contain the name of the

changed data item. In general, it may have extra information about other data items such as timestamps of their most recent changes.

- Synchronous: When the invalidation reports are broadcast periodically.

Asynchronous methods do not provide any latency guarantees for the querying client; if the client queries a data item after disconnection period then it either has to wait for the next invalidation report (with no time bound on the waiting time) or has to submit the query to the server(cache miss). In case of synchronous method, there is a guaranteed latency due to the periodic nature of synchronous broadcast. For stateless synchronous method, two algorithms proposed by Barbara and Imielinksi [6] (i.e., Timestamps (TS) and Amnesic Terminals (AT)) are proposed in the literature.

In the $stateful$ approach, an MSS maintains object state for each MU's cache and only broadcasts IRs for those objects. Kahol, et al. [32] proposed an asynchronous stateful (AS) algorithm which uses asynchronous invalidation reports to maintain cache consistency, i.e., reports are broadcast by the server only when some data changes and not periodically.

In AS, each MU maintains its own *Home Location Cache* (HLC) to deal with the problem of disconnections. The HLC of an MU is maintained at the designated *home* Mobile Switching Station (MSS). If a MU is roaming, its HLC is duplicated at the MSS of its current cell. Thus, an MSS always maintains a HLC for each MU in its coverage area at any given time. Consider an MSS with N mobile users ($MU_i, 1 \leq i \leq N$) at any given time. For any i, HLC_i for MU_i, as maintained in the MSS, keeps track of what data has been locally cached at MU_i (state information of the MU). In general, HLC_i is a list of records $(x, T, invalid\_flag)$ for each data item x locally cached at MU_i, where x is the identifier of a data item and T is the timestamp of the last invalidation of x. The key feature of AS scheme is that the invalidation report are transmitted asynchronously and those reports are buffered at the MSS (in the HLCs of the mobile user) until an

explicit acknowledgment is received from the specific MU. The $invalid\_flag$ in the HLC record for the specific data item is set to TRUE for data items for which an invalidation has been sent to the MU but no acknowledgement has been received. The timestamp is the same as that provided by the server in its invalidation message.

Each MU maintains a local cache of data items which it frequently accesses. Before answer any queries from the application, it checks if the requested data is in a consistent state. When a MSS receive receives an invalidation from a server, the MSS determines the set of MUs that are using the data by consulting the HLCs and sends an invalidation report to each of them. When a MU receives that invalidation message, it marks the particular data item in its local cache to be invalid. When an MU receives (from the application layer) a query for a data item, it checks the validity of the item in its local cache. Of the item is valid, it satisfies the query from its local cache and saves on latency, bandwidth and battery power; otherwise, an uplink request to the MSS for the data item is required. The MSS make a request to the server for the data item on behalf of the MU. When the data item is received the MSS adds an entry to the HLC for the requested data item and forwards the data item to the MU. Note that the data item may or may not be cached at the MSS.

Wang, et al. proposed a hybrid consistency maintenance scheme, called Scalable Asynchronous Cache Consistency Scheme (SACCS) to maintain MU's cache consistency of mobile computing systems [89] [90] [91]. Unlike the stateful algorithm which requires the MSS to remember all data objects for each and every MU's cache, SACCS only requires the MSS to identify which data objects in its database might be valid in MU's caches. This makes the management of the MSS database much simpler. On the other hand, unlike existing synchronous stateless approaches, SACCS does not require periodic broadcast of IRs, thus greatly reducing IR messages that need to be sent through the downlink broadcast channel.

For each cached data object, SACCS uses a single flag bit f_x, to maintain the consistency between the MSS and MU caches. When data item d_x is retrieved by an MU, f_x is set ($f_x = 1$), indicating that a valid copy of d_x may be available in an MU's cache. If and when the MSS receives an updated d_x, it broadcasts an invalidation report $IR(x)$ and resets f_x ($f_x = 0$). This action implies that there are no valid copies of d_x in any MU's caches. Furthermore, while $f_x = 0$, subsequent updates do not entail broadcast of $IR(x)$. The flag f_x is set again when the MSS services a retrieval (including request and confirmation) for d_x by an MU. For every data object with unique identifier x, the data structure for MSS is as follows:

- (d_x, t_x, f_x): data entry format for each data object in MSS. Here d_x is the data object, t_x is the last update time for the data object and f_x is a flag bit.

In mobile environments, an MU's cache is in one of two states: (i) awake or (ii) sleep. If an MU is awake at the time of $IR(x)$ broadcast, the d_x copy is invalidated and an *ID-only* entry is maintained by the MU. The data objects of an MU in the sleep state are unaffected until it wakes up. When an MU wakes up, it sets all cached valid data objects (including d_x) into the *uncertain* state. Consequently, sleeping MUs and the cached object are unaffected by missing $IR(x)$ broadcast.

2.3 Caching and Prefetching

The constraints of mobile wireless environments, such as scarce bandwidth and limited client resources, remain barriers to the full utilization of the capabilities of mobile computing. Client data caching has been considered a good solution for coping with inefficiency of wireless data dissemination because it reduces the amount of traffic over the wireless communication channel by answering queries from data cached at the client. There are three main issues involved in cache design for mobile devices: 1) a cache *replacement* policy which chooses to discard the victim data set currently in the cache

for accommodating new incoming data; 2) a cache *prefetching* policy which automatically prefetches data into cache for possible future accesses; and 3) cache *invalidation* scheme which maintains the data consistency between local cache and the original server. In this section, we investigate first two issues of the mobile data caching.

2.3.1 Caching Replacement Algorithms

Ideally, a cache replacement algorithm is required to achieve multiple goals. The goals may include high hit ratio, low access latency, low stretch, or high energy saving. In order to achieve a specific performance goal, the cache replacement [99] use different parameters to evaluate the importance of cached data items and chooses the victim data set at each replacement. In other words, the cache replacement algorithm should aim to optimize the following expression

$$max \sum_{i \in C} U_j(d_i) \tag{2.1}$$

subject to a constraint

$$\sum_{i \in C} s_i \leq S \tag{2.2}$$

where C is the set of data items selected for caching, S is the total cache size and $U(d_i)$ is the utility value of cached data item d_i for goal j (the goal can be hit ratio, access delay and energy saving etc.). The problem defined by Equations (2.1) and (2.3) is equivalent to the knapsack problem, which is NP-hard. Consequently, there is no known efficient algorithm for solving the problem. However, if we assume that the sizes of cached documents are relatively small when compared with the total cache size S, and thus it is always possible to utilize almost the entire cache space, then the solution space can be restricted to sets of documents C satisfying:

$$\sum_{i \in C} s_i = S \tag{2.3}$$

and there exists some optimal solutions by using simple greedy off-line algorithm.

For the cost consideration, there have been several algorithms developed for the uniform-size variable-cost data items caching problem. GreedyDual [103], is actually a range of algorithms which include a generalization of LRU and a generalization of FIFO. The name GreedyDual comes from the technique used to prove that this entire range of algorithms is optimal according to its *competitive ratio*. The competitive ratio is essentially the maximum ratio of the algorithms cost to the optimal offline algorithm's cost over all possible request sequences. In [15], Cao et al. proposed a variant algorithm, GreedyDual-Size, which handles data items of differing sizes and differing cost. They also showed that GreedyDual-Size has an optimal competitive ratio.

In the literature, there are other cost based algorithms that have been proposed to optimize the documents of web proxy cache in the wired networks. Least-Recently-Used (LRU) evicts the document with was requested the least recently. Least-Frequently-Used (LFU) evicts the document which is accessed least frequently. Size [95] evicts the largest document. LRU-Threshold [1] is the same as LRU, except documents large than a certain threshold size are never cached. Hyper-G [95] is a refinement of LFU with last access time and size considerations. Lowest-Latency-First [96] tries to minimize average latency by removing the document with the lowest download latency first. Hybrid [96] is aimed at reducing the total latency. Lowest Relative Value (LRV) [58] includes the cost and size of a document in the calculation of a value that estimates the utility of keeping a document in the cache.

The cache replacement algorithm of wireless data access was first studied in the *Broadcast Disks* project. Acharya, et al. propose a cache replacement policy called *PIX* [2], 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 has a short retrieval delay. As an example of

PIX, consider two pages. One page is accessed 1% of the time at a particular client and is also broadcast 1% of the time. A second page is accessed only 0.5% of the time at the client, but is broadcast only 0.1% of the time. In this example, the former page has a lower PIX value than the latter. As a result, a page replacement policy based on *PIX* would replace the first page in favor of the second, even though the first page is accessed twice as frequently.

Tassiulas and Su presented a cache update policy that attempted to minimize average access latency [86]. In [86], the broadcast channel was divided into time slots of equal size, which were equal to the broadcast time of a single item. Let λ_i be the access rate for item i and $\tau_i(n)$ be the amount of time from slot n to the next transmission of item i. A time-dependent reward (latency reduction) for item i in slot n is given by $r(i,n) = \lambda_i \tau_i(n) + \frac{\lambda_i}{2}$. The proposed *W-step look-ahead* scheme made the cache update decision at slot n, such that the cumulative average reward from slot n up to slot $n+W$ was maximized. The larger the window W, the better the access performance, but complexity of the algorithm is high.

Caching algorithms for the Broadcast Disks systems were also investigated by Khanna and Liberatore [37]. Different from the previous work, their work assumed that neither knowledge of future data requests nor knowledge of access probability distribution over the data items was available to the clients. The proposed *Gray* algorithm took into consideration the factors of both access history and retrieval delay for cache replacement/prefetching. Theoretical study showed that, in terms of worst-case performance, *Gray* outperformed LRU by a factor proportional to $CacheSize/\log CacheSize$.

Cache replacement policies in a data updating environment were investigated by Xu et al. [100]. Different from the previous work, variable data sizes, data updates are considered in the design of cache replacement algorithm. An cache replacement policy, called *Min-SAUD*, which accounts for the cost of ensuing cache consistency before each

cached item is used. *Min-SAUD* uses *stretch* as the major performance metric, which accounts for the data service time and, thus is fair for data items with various sizes. Based on the optimization of stretch performance, a *gain function* is derived to evaluate each cached data item. The function is given as follows:

$$gain(i) = \frac{p_i}{s_i}\left(\frac{b_i}{1+x_i} - v\right) \tag{2.4}$$

where p_i is the access probability of data item i, b_i is the retrieval delay from the server, x_i is the ratio of update rate to access rate for the data item, s_i is the size of the data item, and v is the cache validation delay.

2.3.2 Prefetching

Prefetching is a technique that can reduce the access latency and improve the cache hit ratio. The idea of prefetching stems from the fact that, after retrieving a page or data item from the remote server, a user usually spends some time viewing the page, and during this period the network link is generally idle. If we can take advantage of this idea by fetching some files that will likely be accessed soon (in other words, prefetching them to the local disk), there will be no transmission delay when the user actually requests these files. In addition, prefetched files can be immediately processed if decryption or decompression is required, which allows further reduction in the delay of loading a page in the mobile device. The difficulty of realizing efficient prefetching lies in the fact that it is impossible to predict exactly what a user is going to need. So some of the prefetched files are never used, which means prefetching increases the system load. At some point, this increase in load may increase the delay of normal (nonprefetching) requests significantly and may eventually cause the overall system performance to degrade. In the wireless network, this also increases the uplink consumption and battery power waste. Therefore,

the key challenge in prefetching is to determine "what to prefetch" so that improvement in performance will be enhanced.

Jiang and Kleinrock [45] proposed an adaptive network prefetch scheme comprising a prediction module and a threshold module. The two modules compute the access probabilities and the prefetch thresholds, respectively. Whenever a new page is displayed, the prediction module updates the local access history, if needed, and computes the access probability of each link on that page. At the same time, for each server which has at least one link on this page, the threshold module computes its prefetch threshold based on network and server conditions as well as the costs of time and bandwidth to the user. Finally, all the files with access probability greater than its server's prefetch threshold are prefetched. The prediction algorithm may also be run at the server, in which case access probabilities will be send to the user along with the requested page.

Speculative prefetching has been proposed to improve the response time of network access. N. J Tuah et al. [87] investigates performance modelling of speculative prefetching. They analyze the performance of a prefethcer that has uncertain knowledge about future access. They also have developed a prefetch algorithm to maximize the improvement in access time. The algorithm is based on finding the best solution to a stretch knapsack problem. An integration between speculative prefetching and caching is also investigated in [87].

2.4 Peer-to-Peer Networks

Recently, with the popularity of peer-to-peer (P2P) file sharing systems, such as Gnutella [105], peer-to-peer computing has drawn much research attention. Much research work has been devoted to provide scalable P2P lookup services, such as Chord [85], CAN [69] and Pastry [71].

With the great success of peer-to-peer networks on the files sharing in Internet, mobile peer-to-peer networks are emerging as a promising mechanism for information sharing in wireless network. Such networks consist of mobile devices called peers, that interact during (brief) physical encounters in the real world, thereby engaging in short-haul wireless exchanges of data [48]. Compared to P2P systems in wired networks, MP2P networks present a more constrained communication environment. Due to limitations of battery power, wireless bandwidth, and the highly dynamic nature of the network topology, MP2P networks give rise to new challenges for research on routing, resource discovery, data retrieval, security and privacy management. Papadopouli and Schulzinne [64] proposed the 7DS architecture, a peer-to-peer data sharing system, in which a couple of new protocols are defined to share and disseminate data among users that experience intermittent connectivity to the Internet. A cooperation concept was introduced in 7DS for data sharing among all mobile hosts. Lim, et al. [53] suggested a cooperative caching scheme for Internet based mobile ad hoc networks. A broadcast based simple search scheme is proposed to establish cooperation among all MUs in the network to share cached data items. Although the broadcast based data search scheme can locate the nearest requested data item, the energy and bandwidth cost of the flooding search is significantly high for a mobile ad hoc network. Sailhan and Issarny [75] limited the broadcast range in collaborative Web caching in ad hoc networks to minimize energy consumption and network load. The authors proposed a fixed broadcast range based on the underlaying routing protocol. However, the mobile users' location, data popularity and network density often change in a real mobile environment, so the fixed broadcast scheme is hard to adapt to real mobile applications. Yin and Cao [102] investigated cooperative caching algorithms in ad hoc networks to support data access. These algorithms mainly focus on the problem of choosing data item or data path for caching in the limited cache space of mobile devices.

In a summary, in order to enable ubiquitous information access to mobile and wireless users, a comprehensive resource aware framework which combines various information access approaches need to be studied. However, most of above works were proposed under some specific application scenarios. Moreover, due to limited resource, network heterogeneity and user mobility, efficient resource utilization, flexibility and scalability are crucial for providing information access to mobile users in mobile and distributed environments. Most related work did not address all these three challenges. In this dissertation, we propose a utility based resource aware framework for information caching and sharing to support ubiquitous information services in mobile and distributed environments. In this framework, *utility functions* are used to optimize cache management and get optimal performances for user data accesses under different wireless networks. The utility functions consider various resource constraints (e.g., energy, bandwidth, memory, network connectivity etc) of mobile and distributed environments. Since different utility functions can adapt to different wireless networks, the framework provides a flexible solution for information services in different application scenarios. Moreover, efficient data consistency maintenance schemes are deployed in the framework to enhance consistent information availability to users. Since the framework efficiently integrates various information providing models, (push-based, pull-based and peer-to-peer based), it can support large scale information services in mobile and distributed environments.

CHAPTER 3

ENERGY-EFFICIENT DATA CACHING AND PASSIVE PREFETCHING

In the next generation mobile systems, cellular networks will continue to serve as the backbone of wireless Internet for providing ubiquitous information services. In our proposed framework, whenever the cellular wireless service is available, mobile users fetch data items from the original servers via base stations using pull based information access. However, limited battery power of mobile terminals and scarce wireless channel bandwidth impose significant challenges to ubiquitous information services. In this chapter, we propose a novel energy and bandwidth efficient data caching mechanism, called GreedyDual Least Utility (GD-LU), that enhances dynamic data availability to the mobile users in cellular wireless networks. The proposed utility-based caching mechanism considers such characteristics of mobile distributed systems as connection-disconnection, mobility handoff, data update and user request patterns to achieve significant energy and bandwidth saving. In mobile and distributed environments, mobile users frequently connect and disconnect from the cellular network. Thus, we need to have some mechanism to guarantee data consistency between the original servers and the caches of mobile users. In our proposed framework, we deploy a scalable asynchronous cache consistency scheme (SACCS) [91] for caches at mobile users and servers to provide consistent information availability in the cellular networks.

In the next generation of mobile systems, digital broadcast networks (DBNs) [36] is also expected to play an important role to provide information services to massive number of mobile users. In digital broadcast networks, a set of information units (Web pages, digital audio or video clips, games) are periodically broadcast (or pushed) over

the broadcast channel. In push based information services environments, the data item requested by one mobile user is available to all other mobile devices in the data dissemination channel. In this chapter, we propose a passive prefetching scheme for cache management of mobile users in push based information service environments. In contrast to existing prefetching schemes, in passive prefetching, mobile users do not send request messages to acquire data item. A threshold (TH) is set at the cache, based on the relative utility value of each data item to admit data appearing at the broadcast channel. This significantly reduces energy and bandwidth consumption for data access in push based information service environments.

3.1 An Overview of SACCS

In order to maintain data consistency between mobile devices and the server, Wang et. al [89] [91] proposed the SACCS algorithm, in which each data item in the server is associated with a flag bit. When an item is retrieved by a mobile device, the corresponding flag bit is set indicating that a valid copy may be available in the mobile cache. When the data item is updated, the server immediately broadcasts its invalidation report (IR) to mobile devices and resets the flag bit. The reset flag bit implies that there is no valid copy in any mobile cache. Hence, subsequent updates do not broadcast IR until the flag bit is set again. A mobile device is either in an *awake* state (i.e., connected with base station) or in a *sleep* state (i.e., disconnected from base station) at the time of IR broadcast. If a mobile device is in the awake state, it deletes the valid copy and sets the entry to *invalidated* state (i.e., the data item is deleted, but an ID is kept) after it receives the IR. If the device is in the sleep state, it misses the IR; and upon wake up, it sets all valid cache entries to an *uncertain* state. An uncertain entry must be refreshed or checked before its usage. Thus, the cache consistency is guaranteed. All entries with data

items in uncertain or invalidated state can be used to identify useful broadcast messages for validating or downloading valid data items.

In SACCS, each data item in the system is in one of three states: *invalidated*, *certain* and *uncertain*. The *invalidated* state is defined for data items that are not cached at the mobile device. The *uncertain* state is defined for data items that are cached at the mobile device, but validation of data items is not confirmed. The *certain* state is defined for data items whose validation is confirmed and can be used to satisfy the application's data request.

After receiving the IR, mobile devices discard the corresponding data items and keep the metadata information (including ID) in their local caches. Also, the data items contained in IR will change their status to the invalidated state. If mobile devices reconnect to wireless networks after a disconnection from the base station for a while or if they move to another cell, all data items in the cache are set to the uncertain state. In both cases, we do not know whether the data in the cache are valid or not. If applications request an item that is in uncertain state, an uncertain request message is initialized and sent to the base station. The base station responds with a confirmation message if the data item has not been updated since the mobile device last acquired it; otherwise, the whole data item is sent to the mobile. After receiving confirmation message from the base station, the uncertain cached data items will be changed to the certain state.

In order to handle IR loss, the server sets an estimated time-to-live (TTL) for each data item based on its update history. This TTL is also cached together with the data item in the mobile device when acquired from the server. The cached item in the mobile device is automatically set to an uncertain state when its TTL expires. This will protect a stale data item from being used for an arbitrarily long time due to IR loss, which means that a connected mobile user cannot correctly receive a broadcast IR. Simulation results reported in [91] show that the performance of SACCS is superior to those of other

existing stateful and stateless cache consistency algorithms in both single and multi-cell wireless environments, we use SACCS in our framework to maintain data consistency of mobile users for accessing information in cellular networks.

3.2 Analytical Model for Utility

In pull based approach to access information in cellular networks, the mobile user continuously requests the original server and retrieves the data from the server through a unicast channel. Four kinds of data management events happen at a mobile user: data request, data update (i.e., receiving data invalidation report), disconnection, and mobility handoff. Figure 3.1 depicts these events for a mobile user. Data update, disconnection, and handoff events may occur zero or more times between two consecutive request events.

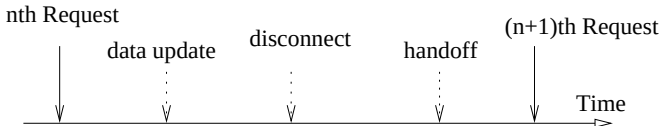

Figure 3.1 Events of Mobile Devices.

Based on the data access model of mobile users in cellular networks, we develop a utility model to analyze the energy saving due to cache usage at a mobile device. First, we state some assumptions used in our analytical model. The outcome of our analysis is a utility function, which will be deployed to manage the cache space of mobile devices.

Since Poisson process has been widely used to model independent arrivals [30], we assume that data request, data update at the server, mobility handoff and disconnection of mobile devices follow Poisson processes. The following analysis assumes a unicast communication between the base station and mobile devices. As mentioned earlier, the

SACCS algorithm is used to maintain data consistency between mobile devices and the original server.

3.2.1 Notations

The notations used in the analysis are listed below:

- d_i: data item i;
- λ_i^u: update rate of data item d_i;
- λ_i^a: access request arrival rate of data item d_i;
- λ^{dc}: disconnection rate of mobile devices;
- λ^h: handoff rate of mobile devices;
- $\lambda^s = \lambda^{dc} + \lambda^h$: uncertain rate of mobile devices;
- λ^a: user request arrival rate of mobile devices;
- $p_i = \lambda_i^a / \lambda^a$: average access probability of d_i;
- s_i: size of data item d_i;
- sup: size of uplink uncertain request or cache missing request;
- sdw: size of downlink confirmation message;
- $\varepsilon(sup)$: energy cost to send a request to a base station;
- $\varepsilon(sdw)$: energy cost to receive a confirmation message;
- $\varepsilon(s_i)$: energy cost to receive a data item of size s_i;
- S_n^u: set of uncertain data items in cache after nth request;
- S_n^c: set of certain data items in cache after nth request;
- $S_n = S_n^u \cup S_n^c$: set of data items in cache after nth request;
- EV_j: event j occurs in mobile device.
- P_j: the probability of event EV_j.

Figure 3.2 depicts the state transitions of each cached data item d_i in the system between nth and $(n+1)$th data requests. Event EV_1 occurs when the mobile device

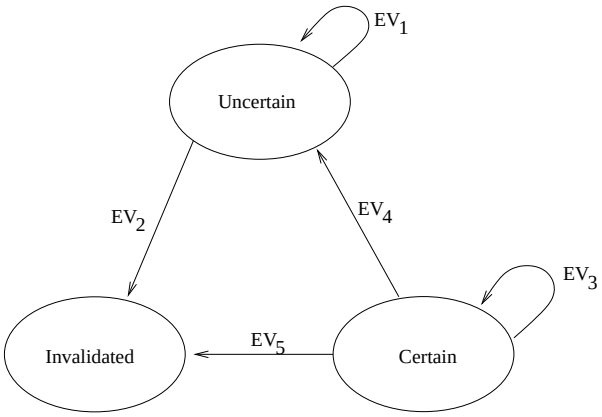

Figure 3.2 States Transition of Mobile Data.

does not receive any IR for d_i which therefore remains in the uncertain state. Event EV_2 occurs when the item d_i moves to an invalidated state as an IR for d_i is received. Event EV_3 occurs if both the following conditions are satisfied: mobile device does not receive any IR for d_i, and no handoff or disconnection occurs. Event EV_4 accounts for d_i changing from certain to uncertain state, if a handoff or disconnection occurs. EV_5 stands for the event that d_i changes to invalidated state from certain state. EV_5 happens if there is at least one IR for d_i received by the mobile device.

3.2.2 Probability of Events

Recall that we assume that the arrivals of data request, data update, disconnection and handoff are all Poisson processes. The probability P_j of each event EV_j (for $1 \leq j \leq 5$) is given by Equations (3.1)-(3.5) respectively.

$$P_1 = \int_0^\infty e^{-\lambda_i^u t} \lambda_i^a e^{-\lambda_i^a t}\, dt = \frac{\lambda_i^a}{\lambda_i^a + \lambda_i^u} \tag{3.1}$$

$$P_2 = 1 - P_1 = \frac{\lambda_i^u}{\lambda_i^a + \lambda_i^u} \tag{3.2}$$

$$P_3 = \int_0^{\infty} e^{-\lambda_i^u t} e^{-\lambda^s t} \lambda_i^a e^{-\lambda_i^a t}\, dt = \frac{\lambda_i^a}{\lambda_i^a + \lambda_i^u + \lambda^s} \tag{3.3}$$

$$P_4 = \int_0^{\infty} e^{-\lambda_i^u t}(1 - e^{-\lambda^s t})\lambda_i^a e^{-\lambda_i^a t}\, dt = \frac{\lambda_i^a}{\lambda_i^a+\lambda_i^u} - \frac{\lambda_i^a}{\lambda_i^a+\lambda_i^u+\lambda^s} \tag{3.4}$$

$$P_5 = 1 - P_3 - P_4 = \frac{\lambda_i^u}{\lambda_i^a + \lambda_i^u} \tag{3.5}$$

3.2.3 Energy Calculation

Energy consumed by any mobile device for sending, receiving or discarding a message is given by the following linear equation [26]:

$$\varepsilon(s) = m * s + h \tag{3.6}$$

where s is the message size, m denotes the incremental energy cost associated with message, and h is the energy cost due to the overheads of message. The latter two parameters, i.e., m and h, are different for sending and receiving. When a mobile user requests an uncertain data item in the cache, two probabilities need to be considered to calculate the energy cost of satisfying user requests. If the requested data item is still valid, the energy cost is $\varepsilon(sup) + \varepsilon(sdw)$, that is the energy costs for sending uncertain request on the uplink and receiving the confirmation message on the downlink. If the requested data item is invalid, the energy cost is given by $\varepsilon(sup) + \varepsilon(s_i)$, the sum of the energy costs for sending uplink uncertain request and for receiving data item. According to Equations (3.1)-(3.5), the energy cost (E_i^u) of the mobile device's request for an uncertain data item $d_i \in S_n^u$ is given by,

$$E_i^u = p_i \left[(\varepsilon(sup) + \varepsilon(sdw)) \frac{\lambda_i^a}{\lambda_i^a + \lambda_i^u} + (\varepsilon(sup) + \varepsilon(s_i)) \frac{\lambda_i^u}{\lambda_i^a + \lambda_i^u} \right]$$

The energy cost (E_i^c) of a mobile device's request for a certain data item $d_i \in S_n^c$ is given by,

$$E_i^c = p_i \left[(\varepsilon(sup) + \varepsilon(sdw))(\frac{\lambda_i^a}{\lambda_i^a + \lambda_i^u} - \frac{\lambda_i^a}{\lambda_i^a + \lambda_i^u + \lambda^s}) + (\varepsilon(sup) + \varepsilon(s_i))\frac{\lambda_i^u}{\lambda_i^a + \lambda_i^u} \right]$$

At the $(n+1)th$ request, the caching mechanism needs to choose victim data set from certain and uncertain data sets in the cache to make space for an incoming data item d_x. Therefore,

$$S_{n+1} = S_n \cup d_x - V_n^u - V_n^c$$

where $V_n^u \in S_n^u$ is the victim data set chosen from the uncertain data set and $V_n^c \in S_n^c$ is the victim data set chosen from the certain set.

So the energy cost after the $(n+1)$th request can be expressed in a recursive way as follows:

$$\begin{aligned} e_{n+1} &= e_n + [E_x^c - p_x(\varepsilon(sup) + \varepsilon(s_x))] \\ &+ \textstyle\sum_{d_i \in V_n^u} [p_i(\varepsilon(sup) + \varepsilon(s_i)) - E_i^u] \\ &+ \textstyle\sum_{d_i \in V_n^c} [p_i(\varepsilon(sup) + \varepsilon(s_i)) - E_i^c] \end{aligned} \tag{3.7}$$

The objective of our replacement policy is to choose the optimal victim data set from the cache to minimize the energy cost after the $(n+1)$th request. The second term in Equation (3.7) is the increased energy cost as d_x is cached at the $(n+1)$th request, i.e., d_x changes from invalidated state to certain state. The first two terms of Equation (3.7) are unaffected by the caching mechanism, whereas the last two terms of the equation can be minimized as follows:

$$\sum_{d_i \in V_n^u} [p_i(\varepsilon(sup) + \varepsilon(s_i)) - E_i^u] = \sum_{d_i \in V_n^u} p_i(\varepsilon(s_i) - \varepsilon(sdw)) \times \frac{\lambda_i^a}{\lambda_i^a + \lambda_i^u}$$

$$\sum_{d_i \in V_n^c} [p_i(\varepsilon(sup) + \varepsilon(s_i)) - E_i^c] = \sum_{d_i \in V_n^c} [p_i(\varepsilon(s_i) - \varepsilon(sdw)) \times \frac{\lambda_i^a}{\lambda_i^a + \lambda_i^u} + (\varepsilon(sup) + \varepsilon(sdw))\frac{\lambda_i^a}{\lambda_i^a + \lambda_i^u + \lambda^s}]$$

Therefore, the utility function to evaluate uncertain data items in a cache is given by,

$$U_i^u = p_i[\varepsilon(s_i) - \varepsilon(sdw)]\frac{\lambda_i^a}{\lambda_i^a + \lambda_i^u} \tag{3.8}$$

The utility function for certain data items is given by,

$$U_i^c = p_i[\varepsilon(s_i) - \varepsilon(sdw)]\frac{\lambda_i^a}{\lambda_i^a+\lambda_i^u} + p_i[\varepsilon(sup) + \varepsilon(sdw)]\frac{\lambda_i^a}{\lambda_i^a+\lambda_i^u+\lambda^s} \tag{3.9}$$

In order to minimize the energy cost at mobile devices, at each replacement, we choose the data items with the least utility values according to Equations (3.8) and (3.9). An uncertain data item costs more energy as it is necessary to get confirmation from the base station. The utility of a data item d_i while it is in the certain state is more than its utility value when in uncertain state by an additive factor of $p_i[\varepsilon(sup)+\varepsilon(sdw)]\frac{\lambda_i^a}{\lambda_i^a+\lambda_i^u+\lambda^s}$.

3.3 GD-LU Cache Replacement Algorithm

The GreedyDual (GD) cache replacement is an efficient online optimal algorithm [103] devised to deal with systems that exhibit heterogeneous data retrieval costs. GD in essence allows a bias to be applied to each item in cache so as to give higher priority to items that incur higher retrieval costs. Revised versions of GD algorithm have been deployed for web proxy caching [14] and multimedia stream caching [49]. Based on the GD concept, we propose a novel algorithm, called *GreedyDual Least Utility* (GD-LU) cache replacement algorithm for mobile devices. Since the utility functions derived from Equations (3.8) and (3.9) are used in the GD-LU algorithm, the parameters in the utility functions should be available when a replacement occurs at the cache. We associate each data item with a metadata that contains the necessary parameters of that item. Parameter estimation is discussed later in Section 3.5. Since each metadata contains history information of the corresponding data item, we use a queue to store the metadata of replaced data items.

C: the set of data items in cache;
u_i: utility value of cached item d_i;
M_i: response message to request data d_i;
L: minimal utility in the cache;
initialize $L = 0$;
user requests a data item d_i
case 1: d_i is in cache and valid
    $u_i = L + U_i^c/s_i$; /* Utility calculation */
case 2: d_i is in cache and uncertain
    send uncertain message to the server;
case 3: d_i is not cached
    send cache missing message to the server;
Wait: message M_i appears at downlink channel;
if M_i is a confirmation message **then**
    set the state of d_i as valid;
    $u_i = L + U_i^u/s_i$;
    return d_i to the application;
if M_i contains the data item d_i **then**
    while there is not enough space for M_i;
      let $L = MIN_{d_k \in C}(u_k)$;
      Evict d_j such that $u_j = L$;
      Keep metadata of d_k in metadata queue;
    end while
    Add d_i into cache;
    $u_i = L + U_i^c/s_i$;

Figure 3.3 GD-LU Cache Replacement Algorithm.

As shown in the algorithm described in Figure 3.3, when an application requests for a data item, the algorithm first checks the state of the item in the cache. If the data item is valid, it is returned and the corresponding metadata is updated. If the data item is in an uncertain state, an uncertain message is sent to the server to check if the data is valid or not since the last retrieval. In the event of a cache miss, the message is sent to the server to retrieve the data. When the mobile device receives the confirmation message, the data item is set to the certain state and returned to the application. If the

whole data item is received, the GD-LU replacement algorithm chooses the victim data set to make space to accommodate the incoming data item.

3.4 Passive Prefetching Algorithm

In the next generation of mobile systems, digital broadcast networks (DBNs) [36] will play an important role to provide information services to massive number of mobile users. In digital broadcast networks, a set of information units (e.g., Web pages, digital audio or video clips, games) are periodically broadcast over the broadcast channel. Mobile users listen to the broadcast channel and acquire their requested data items. DBNs are able to efficiently reach large groups of subscribers and provide them with a high-rate, through one way data broadcast services.

In push based information services environments, the data item requested by one mobile user is available to all other mobile devices in the data dissemination channel. Although the data item may not be requested by a mobile user, prefetching data into the cache of that mobile device may reduce future access latency. However, acquiring the data item from dissemination channel still costs energy at the mobile users. Therefore, blind prefetching of the data that a user never requested may result in wasting energy and replacement of some recently accessed data, thus degrading cache performance.

In order to avoid blind prefetching, we propose a passive prefetching scheme for cache management of mobile users in push based information service environments. In passive prefetching, a threshold (TH) is set at the cache to admit data appearing at the broadcast channel. According to the GD-LU replacement algorithm, a data item whose metadata are kept in the metadata queue may have a higher probability to be requested again by the mobile device. So the data item whose metadata is kept in the metadata queue is considered as valuable candidate for passive prefetching. To evaluate the future

utility of an item at the mobile device, we define *relative utility* (RU) as the ratio of the utility of data (d_i) to the utility of cached data. Thus,

$$RU(d_i) = \frac{U_i^c(d_i)}{MAX_{d_j \in C}(u_j) - MIN_{d_j \in C}(u_j)} \tag{3.10}$$

where C is the set of data items in cache, and u_j is the utility value of cached data item d_j. If RU is greater than the threshold (TH) of the cache, the data item is admitted into the cache; otherwise the cache ignores such data. Although the utility function in Equation (3.9) is derived based on a unicast assumption, as shown in the next section, it can adapt to broadcast communication very well. In the later part of performance evaluation section, the threshold (TH) will be discussed in more details. A median utility threshold scheme is proposed to achieve a near-optimal performance tradeoff between access latency and power consumption. The passive prefetching algorithm is shown in Figure 3.4. In passive prefetching, a mobile device does not send uplink request, thus reducing the burden on the server while improving on cache performance at the mobile device and saving scarce wireless bandwidth.

TH: prefetch threshold set at mobile devices.
for each data item d_f appearing at broadcast channel
  if d_f is in metadata queue **then**
    Calculate $RU(d_f)$;
    if $RU(d_f) > TH$ **then**
      while there is not enough space for d_f;
        let $L = MIN_{d_j \in C}(u_j)$;
        Evict d_j such that $u_j = L$;
        Keep metadata of d_j in metadata queue;
      end while
      Add d_f into cache;
      $u_f = L + U_f^c / s_f$;

Figure 3.4 GD-LU Passive Prefetch Algorithm.

3.5 Implementation Issues and Complexity

To implement the proposed replacement and passive prefetching schemes at the mobile devices, two issues must be addressed; namely, the data structure management and the parameter estimation. Like the GreedyDual algorithm [103], the GD-LU mechanism also uses a priority queue for the data items based on their utility values. Handling a cache hit requires $O(\log N)$ time, where N is the number of data items in the cache. Handling an eviction also requires $O(\log N)$ time. For both cache hit and eviction, the GD-LU replacement requires only one data item to be updated. Due to the priority queue management, the proposed passive prefetching algorithm also has a time complexity of $O(\log N)$.

Several parameters are involved in computing the utility value of a data item. The incremental energy coefficient m and overhead h in Equation (3.6) are system parameters obtained from system specifications. The disconnection rate (λ^{dc}), handoff rate (λ^{h}) and request arrival rate (λ^{a}) can be estimated by using the sliding average method [82]. The update rate (λ_i^{u}) and access rate (λ_i^{a}) of each data item can be estimated by the exponential aging [82] or sliding average method.

We use a metadata to keep the information (e.g., λ_i^{a} and λ_i^{u}) of each data item. For each eviction, the metadata of victim items are kept in the metadata queue. The memory space overhead of each metadata is very small, and the total space overhead of our proposed cache mechanism is $smd * (N + Q)$, where smd is the size of one metadata, N is the total number data items in the cache, and Q is the maximum length of the metadata queue.

3.6 Performance Evaluation

In this section, we evaluate the performance of the proposed GD-LU replacement and passive prefetching algorithms under various system settings and environments. Different performance metrics are used to evaluate three main aspects concerned with the applications (e.g., energy efficiency, access latency and data availability). The Min-SAUD [100], GreedyDual-Size (GD-Size) [14] and traditional Least Recently Used (LRU) algorithms are included for comparison in the performance evaluation study.

3.6.1 Simulation Model and System Parameters

For web data, the size distribution follows the lognormal model [7]. In our simulation also, the lognormal model is used for the size distribution of all data items in the system. We consider a total of 10,000 data items. The cutoff minimal size of the data item is assumed to be 256 bytes, the cutoff maximal size is 80K bytes, and the average size is 25.7K bytes. For each data item, the period between two updates follows an exponential distribution. At original settings, the mean update period of each data item is chosen from a set of {100, 200, 400, 800, 1600, 3200, 6400, 12800, 25600, 51200} seconds with equal probability.

The arrival of requests for each mobile device is assumed to follow a Poisson process with a mean arrival rate of $1/10sec^{-1}$. We consider *Zipf-like* [10] distribution for mobile device's access pattern, in which the access probability p_i of data item d_i is proportional to its popularity rank, *rank(i)* where rank is 1 for most popular item. That is, $p_i = c/rank(i)^{\theta}$, where c is the normalization constant and θ is a parameter between 0 and 1 whose default value is 0.9 in simulations. A sleep-wakeup process [6] is used to model the connection and disconnection behavior of mobile devices. In this model, the arrival rate of the event that a connected mobile device goes into the disconnected state is $\alpha = 1/(1-r)T_s$ and the arrival rate of the event that a disconnected mobile device goes

into the connected state is $\beta = 1/rT_s$. T_s is the duration of a connected-disconnected cycle whose default setting is 3600 seconds, and r is the ratio of the connected time to the connected-disconnected period whose default value is 0.4. We use the linear power consumption model for mobile devices [26]. One broadcast channel of 2 Mbps bandwidth is shared by all mobiles. In the simulation, there are 180 mobile devices, each having a cache of size 2M bytes. The size of a request message (i.e., uncertain and cache miss) is 20 bytes. The size of confirmation and IR message is 16 bytes.

3.6.2 Performance Metrics

The performance metrics considered are the hit ratio (HR), energy per query (EPQ), stretch (ST) [42] and energy stretch (EST) [101]. The metric EPQ is defined as the ratio of the total energy consumed by data requests to the number of requests in a given period. ST is defined as the ratio of the response time of a request to its service time, where service time is defined as the ratio of the requested item size to the bandwidth. EST is defined as the product of stretch and the energy consumption of a data request. Clearly, energy stretch reflects a performance tradeoff between energy consumption and access latency.

3.6.3 Simulation Results

In the experiments, we evaluate the performance of GD-LU replacement algorithm and compare it with the LRU, GD-Size [14], and Min-SAUD [100] algorithms. For performance comparison, two versions of the GD-Size algorithm are used, namely, GD-Size(1) and GD-Size(packets) [14]. The cost of each document is set to 1 for GD-Size(1), and $2 + size/536$ which is the estimated number of network packets sent and received if a miss to the document occurs for GD-Size(packets). As Min-SAUD was originally proposed with the time stamp (TS) based cache consistency maintenance, to ensure

fair comparison, we implement the Min-SAUD under SACCS consistency maintenance scheme. For this, we set the cache validation delay (v) to zero for valid data items and set v to the confirmation delay for uncertain data items as we calculate the gain function of Min-SAUD. In the following figures, for brevity, *SC* stands for the SACCS cache consistency scheme, *MIN* stands for the Min-SAUD algorithm, *GD-SIZE(1)* and *GD-SIZE(PKT)* stand for the two versions of the GD-Size algorithm.

3.6.3.1 Passive Prefetching

In the passive prefetching, a large threshold (TH) value means few data items to be prefetched. As the threshold decreases, more and more data items can be prefetched. When the threshold is zero, all data items in the downlink channel are prefetched. Figures 3.5-3.8 show the performance of EPQ, HR, ST and EST metrics against prefetch threshold under different cache sizes, respectively.

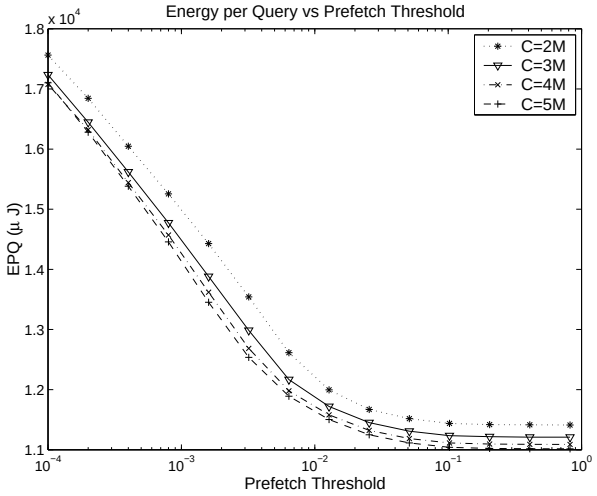

Figure 3.5 Energy Consumption with Passive Prefetching.

As shown in Figure 3.5, the energy consumption is high when the threshold (TH) value is small. For $TH \leq 10^{-2}$, the energy consumption significantly decreases as the

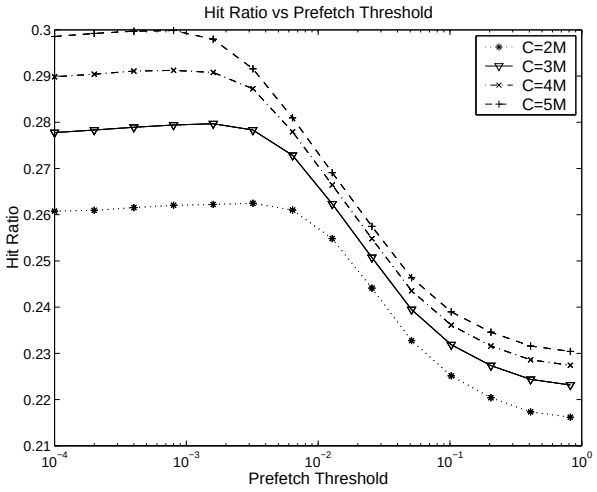

Figure 3.6 Hit Ratio with Passive Prefetching.

threshold increases. But if the threshold is more than 10^{-2}, the energy consumption is almost a constant. This is because more data items are prefetched when the threshold is set at a lower value and some of them are not useful to users. When the threshold is large enough, most of the prefetched data items are useful thus leading to constant energy consumption. Figure 3.6 shows that the hit ratio improves as the threshold decreases. However, at a low threshold of $TH = 10^{-3}$, the hit ratio does not improve although more data items are prefetched. This is called *blind prefetching*. A similar behavior is observed in stretch performance in Figure 3.7. The stretch of access drops with decreasing threshold, since prefetched data items help to improve the hit ratio and reduce stretch. However, blind prefetching does not improve stretch performance due to more energy consumption without any improvement in cache performance. From Figures 3.5, 3.6 and 3.7, we also conclude that devices with more cache space have less energy consumption, and enjoy high hit ratio and less access latency than devices with less cache space.

Since energy consumption of passive prefetching is determined by the threshold value, the passive prefetching scheme can dynamically adjust the threshold according to the current energy level of the devices. When battery power is not a big concern, we

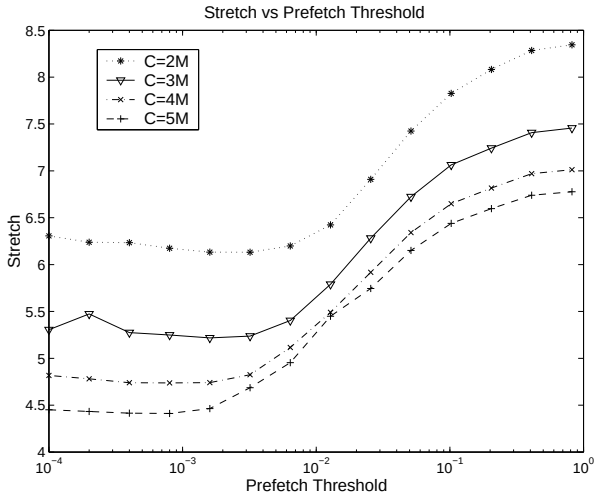

Figure 3.7 Stretch with Passive Prefetching.

Table 3.1 Median Utility Prefetching Performance

Cache Size	EPQ($10^{-2}Joule$)	HR	ST	EST ($10^{-2}Joule$)	Min EST
2MB	1.20368	0.256328	6.39952	7.70297	7.7069
3MB	1.2057	0.271676	5.45998	6.5831	6.57452
4MB	1.21802	0.281747	4.99865	6.0845	6.12083
5MB	1.23306	0.289208	4.75577	5.86415	5.87524

set the threshold at a low value to reduce data access latency. When a mobile is at low energy level, we choose high threshold to save the battery power. However, the setting of threshold should not result in blind prefetching.

As seen in Figure 3.8, if we consider EST as the main performance metric for passive prefetching, we can get an optimal value of threshold. Since energy stretch is the metric that factors in both energy consumption and access latency, the optimal value of threshold can be considered as the *best point* for the tradeoff between energy consumption and access latency.

As shown in Table 3.1, if we choose the points with minimal energy stretch (Min EST) from Figure 3.8 for comparison, the energy stretch performance of median utility

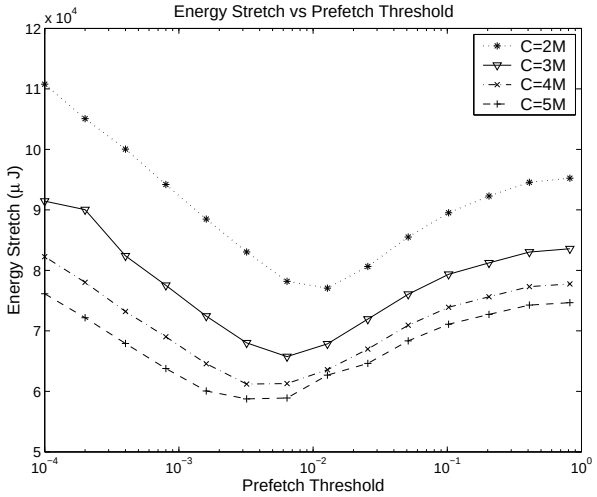

Figure 3.8 Energy Stretch with Passive Prefetching.

threshold setting (EST) achieves near-optimal performance under different cache sizes. This demonstrates that the median relative utility threshold setting is a good heuristic method to achieve optimal performance tradeoff between energy consumption and access latency.

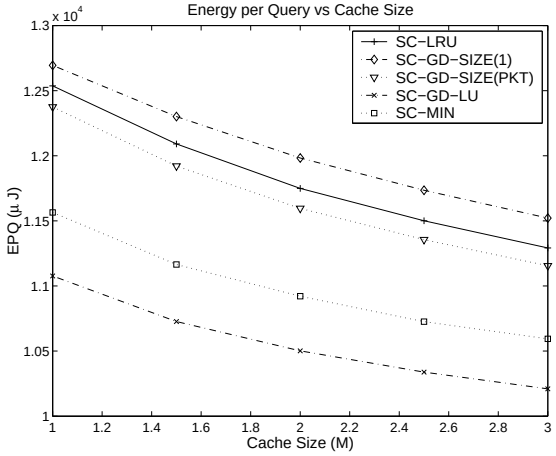

Figure 3.9 Energy Consumption under Different Cache Sizes.

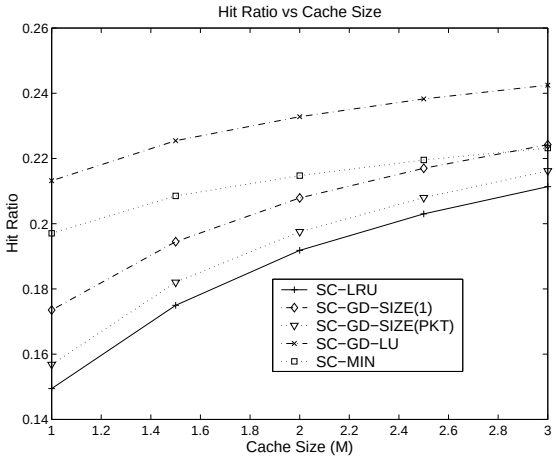

Figure 3.10 Hit Ratio under Different Cache Sizes.

3.6.3.2 GD-LU Replacement

(a) Effect of Cache Size: In this experiment, we evaluate the performance of GD-LU replacement under different cache sizes using EPQ, HR and ST as the performance metrics. In order to evaluate only the cache replacement algorithms, we do not consider passive prefetching scheme in this experiment. As shown in Figure 3.9, power consumption at mobile devices decreases with increasing cache size. GD-LU achieves best EPQ performance among all schemes, and GD-Size(1) exhibits worst EPQ performance. Both Min-SAUD and GD-LU outperform the LRU algorithm in terms of energy consumption, since both of them significantly improve the hit ratio as shown in Figure 3.10. However, the improvement on energy saving of GD-LU is about 15% which is larger than that of Min-SAUD. Because the goal of Min-SAUD is to improve the stretch performance, data items with small size are preferred since smaller size data items contribute more to the stretch performance. In contrast, GD-LU endeavors to achieve optimal energy conservation, therefore, the data items that consume more energy are selected to be cached. Thus, GD-LU outperforms Min-SUAD in terms of energy consumption. Furthermore, GD-LU outperforms both GD-Size(1) and GD-Size(packets), since GD-Size does not con-

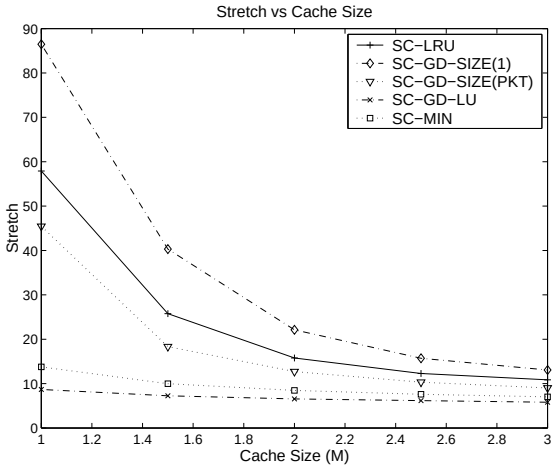

Figure 3.11 Stretch under Different Cache Sizes.

sider data update in the cost function. The EPQ performance of GD-Size(1) is worse than LRU, since GD-Size(1) only tries to minimize the miss ratio by giving preference to small data items.

In Figure 3.10, LRU shows worst hit ratio (HR) performance, and GD-LU shows best hit ratio performance. Min-SAUD performs better than GD-Size(1) for small cache sizes, while Min-SAUD achieves a similar hit ratio performance as GD-Size(1) at large cache sizes. When the cache size is small, the hit ratio is very sensitive to the total number of data items in the cache. The algorithm favoring small data items can cache more data items, so Min-SAUD achieves better HR performance than GD-Size(1) which achieves better HR performance than GD-Size(packets). This is because GD-Size(1) tries to minimize the miss ratio, while GD-Size(packets) tries to minimize the network traffic. In Figure 3.11, both GD-LU and Min-SAUD achieve similar stretch performance which is a significant improvement compared to GD-Size(1), LRU and GD-Size(packets).

(b) Effect of Data Update Rate: As mentioned at the beginning of this section, the default data update rates have a uniform distribution. The mean update period is 6855 seconds for all data items in the system. In this experiment, the mean update

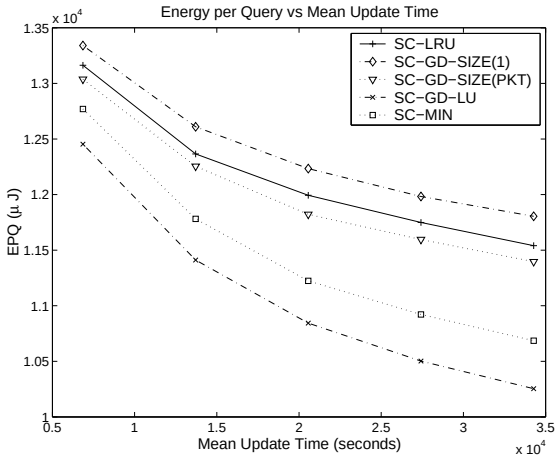

Figure 3.12 Energy Consumption under Different Data Updating Rates.

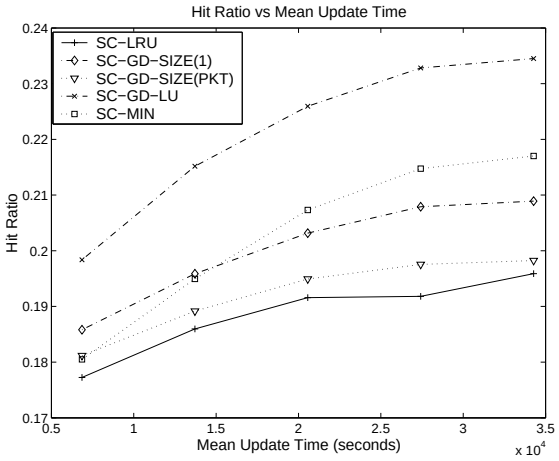

Figure 3.13 Hit Ratio under Different Data Updating Rates.

period varies. As shown in Figure 3.12, the energy consumption decreases as the mean update period increases for all algorithms. There are two reasons for the EPQ dropping as the mean update period increases. First, SACCS broadcasts fewer IR messages as the mean update period increases, so the energy used for IR listening is less at a low data update rate. Second, as seen in Figure 3.13, the hit ratio performance increases as the mean update period increases, and each scheme saves more energy by reducing cache misses. As shown in Figure 3.12, GD-LU achieves best EPQ performance among

all schemes, but the improvement from LRU scheme is low at a high update rate. When the data update rate is high, each mobile device needs to listen to many IR messages, thus IR listening dominates the energy consumption for mobile devices. This leads to low EPQ improvement of GD-LU compared to LRU at high update rates. But the EPQ improvement of GD-LU from LRU is higher than that of Min-SAUD at different updating rates, and GD-LU also outperforms two versions of the GD-Size algorithm.

As shown in Figure 3.13, all algorithms achieve better hit ratio performance in a lower update rate environment. The hit ratio performance of GD-LU is better than that of other schemes under different update rates. Min-SAUD achieves similar hit ratio performance as LRU at a high update rate. This is because the gain function used in Min-SAUD approaches zero for each data item at high data update rates. Thus, Min-SAUD is degraded as LRU, which only evaluates recent access.

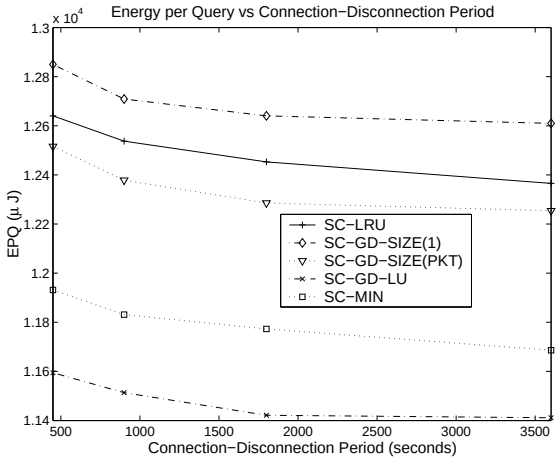

Figure 3.14 Energy Consumption under Different Connection/Disconnection Period.

(c) Effect of Disconnection Frequency: This experiment evaluates the performance of GD-LU under different disconnection frequencies. The connection-disconnection duration (T_s) is varied to simulate the different disconnection frequencies of mobile de-

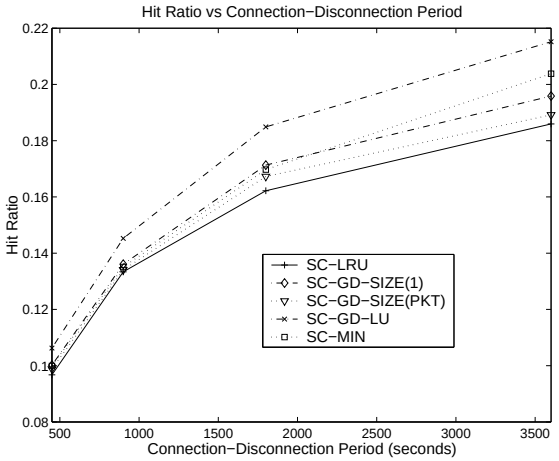

Figure 3.15 Hit Ratio under Different Connection/Disconnection Period.

vices. A small value of T_s indicates a high disconnection frequency, and vice versa. Figure 3.14 shows that GD-LU has a significant improvement on EPQ performance over LRU under different disconnection frequency, and also outperforms Min-SAUD and GD-Size(packets). GD-Size(1) shows worst EPQ performance under different disconnection frequencies. According to the SACCS algorithm, more data items in the cache are in uncertain state as the disconnection frequency increases. Since the hit ratio is counted only for valid data, as shown in Figure 3.15, LRU, GD-Size(1), GD-Size(packets) and Min-SAUD achieve similar hit ratio performance for high disconnection scenario, while GD-LU gets best hit ratio performance under different disconnection frequencies. This is because the utility function deployed in GD-LU considers connection-disconnection characteristic of mobile devices. Figure 3.16 shows that the GD-LU achieves best stretch performance among all schemes under different connection-disconnection periods.

3.7 Summary

In this chapter, we investigate two data access mechanisms of the proposed framework under two different wireless service networks. First, we study the pull based in-

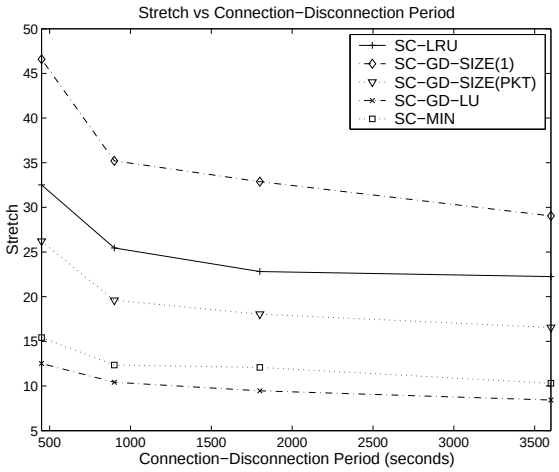

Figure 3.16 Stretch under Different Connection/Disconnection Period.

formation access in wireless cellular networks. In order to save energy and wireless bandwidth and reduce access latency, we propose a novel caching mechanism for pull based information access. A scalable asynchronous cache consistency scheme (SACCS) is deployed at mobile users and the servers to maintain data consistency. A utility model is built to investigate the energy conservation by deploying a cache at mobile devices in the cellular networks. Based on the utility function derived from the analytical model, we propose a novel caching mechanism, called GreedyDual Least Utility (GD-LU). Simulation results show that GD-LU achieves more than 10% energy saving for mobile users in pull based information access in cellular networks.

Second, we study the push based information access in digital broadcast networks. In order to save energy and reduce the latency of information accesses, a passive prefetching scheme is proposed to predict users' future information accesses based on relative utility value of data items. Simulation experiments demonstrate that the passive prefetching algorithm can achieve near-optimal performance tradeoff between access latency and energy consumption by choosing proper threshold.

Both pull based and push based information access approaches rely on client server model to provide the information services to mobile users, and employ wireless infrastructure and an information server. However, in some application scenarios, the wireless infrastructure may be inaccessible. In order for mobile users to continuously access information in such scenarios, we propose peer-to-peer information access approaches under different wireless environments.

In Chapter 4, we study a peer-to-peer caching approach to help mobile users access information when they are not in the service coverage of wireless infrastructure. In Chapter 5, a location-aided peer-to-peer data search scheme is proposed to further reduce energy and bandwidth consumption of each data access of mobile users in the mobile environments without wireless infrastructure.

CHAPTER 4

PEER-TO-PEER CACHING IN MULTI-HOP HYBRID NETWORKS

In the last chapter we studied pull and push based information access approaches under cellular networks and digital broadcast networks. However, in some application scenarios, mobile users are temporarily out of the service coverage of the wireless infrastructure, or the wireless infrastructure is temporarily not available due to network congestion. In order to enable ubiquitous information services in such scenarios, mobile ad hoc networks (MANET) is considered as complementary to wireless infrastructure (e.g., cellular or wireless LANs). Indeed, hybrid wireless networks as an integration of infrastructure based and mobile ad hoc networks are emerging as an attractive solution to providing ubiquitous information services. In such networks, mobile users not only can access information through cellular networks but also can retrieve information from other mobile users via multi-hop peer-to-peer communications.

In this chapter, we introduce a novel scheme called *energy-efficient peer-to-peer caching with optimal radius* (EPCOR), to help mobile users to efficiently access information in the hybrid wireless networks. In this scheme, a peer-to-peer (P2P) overlay network is built among the mobile users (MUs) to facilitate cooperative data sharing. In particular, for energy conservation, each MU in the P2P overlay shares a data item in a *cooperation zone*. An analytical model is developed to evaluate the performance improvement due to energy conservation in the EPCOR scheme. An algorithm is developed to determine the optimal radius of the cooperation zone, based on the trade-off between performance improvement and the overhead of cooperation.

4.1 Hybrid Wireless Networks

Before proposing the EPCOR scheme, let us fist give a brief introduction to hybrid wireless networks. To provide wireless Internet services, currently there exist two major categories of *infrastructure-based* networks: cellular and Wi-Fi networks (e.g., IEEE 802.11 based wireless LANs). Cellular networks operate on licensed frequencies and offer wide geographic coverage (up to 20 Km) but limited bit rate bandwidth (e.g., 107.2 Kbps for GPRS). On the other hand, Wi-Fi networks operate on unlicensed frequencies and offer higher bit rate (e.g., 11 Mbps for IEEE 802.11b and up to 54 Mbps for IEEE 802.11g) but very limited coverage (up to 250 m). In order to provide mobile data services efficiently, both high bit rate and wide coverage are desirable. For this purpose, *ad hoc* networks are considered as complementary to infrastructure-based networks. Communication networks for such scenarios as battle field and disaster relief operations are examples of ad hoc networks. In ad hoc networks, each MU retrieves the desired information from other MUs through multi-hop paths in which MUs cooperatively act as routers to relay the data traffic. However, due to the absence of stable network connection, a pure ad hoc network has a low reliability.

Recently, several models have been proposed for designing *hybrid networks* that integrate ad hoc and infrastructure-based networks with a goal to extend the coverage area of Wi-Fi networks [34] [55] [56] [106] and also improve the throughput of cellular networks [41] [59] [98]. In hybrid networks, the base stations (or access points) are assumed to be attached to the Internet or an infostation [38]. When an MU, say A, moves out of the transmission range of its base station (BS) or if the quality of cellular wireless link is low, other MUs along the path to the BS allow A to continue to access information by relaying its packets to the BS. The proxy MUs around the BS serve as a bridge to exchange data between the BS and other MUs. Figure 4.1 illustrates an example of a hybrid wireless network. Since the hybrid networks significantly improve

the data rate and extend the wireless service coverage area at low cost, this network model offers an attractive solution to providing mobile data services.

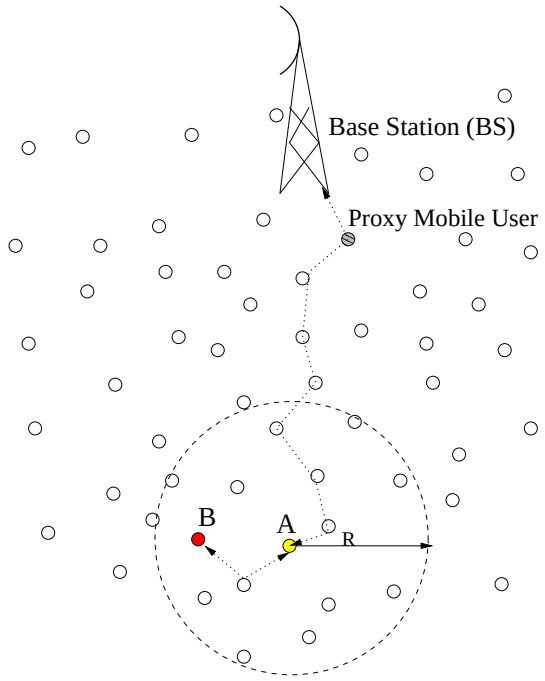

Figure 4.1 An Example of Multi-Hop Hybrid Wireless Networks.

4.2 Description of EPCOR Scheme

Each mobile user in a hybrid network has dual communication capability. If an MU is within the transmission range of a base station, it accesses the information directly from the BS in a single hop; otherwise the MU uses peer-to-peer ad hoc links to retrieve data through multiple hops. Let the BS be a source of all items. The BS may retrieve the data items from the Internet through a wired network, or from an attached infostation. A data query initiated by an MU is sent to the BS along with the routing path; upon receiving the request, the BS responds with the requested data item. A multi-hop routing protocol (e.g., [39], [65], [66]) is assumed to route data packets in the network.

In the EPCOR scheme, a peer-to-peer (P2P) overlay network is created among the mobile users to facilitate cooperative sharing of data. Two peers in the overlay network periodically exchange their cache index messages to maintain their neighborhood relationship in the overlay. In order to conserve energy, the message exchange is localized in a cooperation zone. We define the *cooperation zone* (CZ_{ij}) of an MU, say MU_i, for sharing the data object d_j as a set of MUs in the overlay. Each MU that belongs to CZ_{ij} receives the cache index message, including the index of d_j from MU_i. The *radius* of a cooperation zone is defined as the maximum number of hops between MU_i and any other MU in the cooperation zone. In the EPCOR scheme, each MU uses a resource table to maintain the indices of the data items cached at neighboring MUs in the P2P overlay. For a cache miss, the MU looks up the resource table and forwards the query to a neighbor that has the requested data item. If more than one neighbor has a copy of the requested item, then the one that entails the least cost of data retrieval is chosen. With reference to Figure 4.1, when a data query is initiated in an MU, say A, it first looks for the data item in its own cache. If there is a cache miss, A checks if the item is cached in other neighboring peers, say B, in the P2P overlay; then the query is forwarded to B. Since the number of hops between A and B is less than that between A and the BS, there is a saving of bandwidth and energy in retrieving the data item.

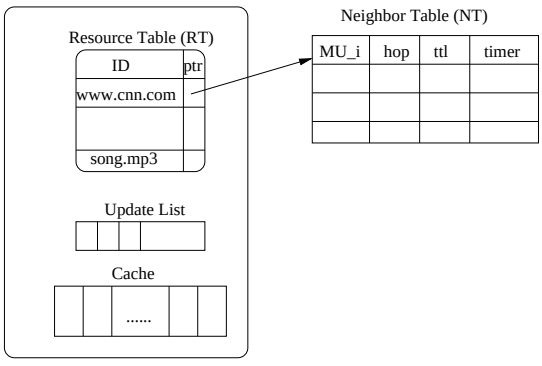

Figure 4.2 Data Structures at a Mobile Node.

4.2.1 Data Structures

Figure 4.2 shows the main data structures maintained in each peer in the EPCOR system. These data structures are resource table (RT), neighbor table(NT), update list and cache. They are formally described below along with message formats:

- $CL = \{d_1, d_2, ..., d_n\}$: set of cached data items.
- $UIM = \{< id_1, htl_1, ttl_1 >, ..., < id_n, htl_n, ttl_n >\}$: update index message where id_j is index of data item d_j; htl_j is its hop-to-live value; ttl_j is the time to live value used to indicate d_j's validation.
- $L_{upd} = \{< id_1, flag >, ..., < id_n, flag >\}$: update list to record the updated data items since the last UIM exchange, where $flag$ bit indicates the states of the item (i.e., 0 for deleted item, 1 for added item).
- $RT_j =< id_j, ptr >$: the entry of resource table for data item d_j where ptr is the pointer to a neighbor table.
- $RT = \{RT_1, RT_2, ..., RT_n\}$: resource table containing the index information for data items cached in neighboring peers;
- $NT_j = \{< MU_1, hop_1, ttl_1, timer_1 >, ..., < MU_n, hop_n, ttl_n, timer_n >\}$: the neighbor table of RT_j, where hop_i is the number hops of the neighboring peer MU_i from the current peer; ttl_i is the time to live value of the data item d_j cached at MU_i; $timer_i$ is the aging timer for the entry of MU_i.

In the EPCOR scheme, each MU maintains a resource table for the cache index information of all neighbors in the overlay. For each entry of a data object in the resource table, we maintain a neighbor table which includes the information of neighboring peers in the cooperation zone that have cached this data object. In order to handle mobility of MUs, we use a *timer* in the each entry of neighbor table to indicate whether the neighboring peer currently resides in the cooperation zone or not.

4.2.2 Peer-to-Peer Overlay Maintenance

The P2P overlay of EPCOR scheme is maintained through periodic exchange of update index messages (UIM) among mobile peers. When a mobile peer successfully fetches (or evicts) a copy of an item, it records index information of the data items in the update list, L_{upd}. In one UIM exchange cycle, the mobile peer creates and disseminates a UIM message according to the information in L_{upd}. Peers share each data item in a cooperation zone. With each data item's index included in the UIM, we associate a *hop-to-live* (*htl*) value to indicate the radius of cooperation zone of the corresponding data item. A time to live (*ttl*) value is also included to indicate the estimation of valid time period of the corresponding item. When an MU, say MU_i, receives a UIM from a source, MU_s, the *htl* value of each entry in UIM is re-calculated. If the *htl* value of an entry reaches zero, it is deleted from the UIM to indicate that the UIM has reached the boundary of the cooperation zone for that data item in the overlay. The calculation of optimal cooperation radius is discussed in Section 4.3. In EPCOR, each MU maintains a resource table for the cache information of all neighbors in the overlay. After receiving a UIM from the source MU_s, for each entry with $ttl > 0$ in the UIM, the MU_i checks if the resource table already has an entry for that data item. If not, a new resource entry is created and inserted in the resource table and an entry of MU_s is included in the neighbor table. If the resource entry is already created at the resource table, the source MU is added to the neighbor table. If the neighbor table is full, the neighbor with lowest value of the ratio ttl/hop or with expired *timer*, is deleted from the neighbor table. For each entry with $ttl = 0$ in the UIM, the corresponding entry of the source MU_s in the neighbor table is deleted, because the data item cached in the MU_s is invalidated. In order to handle mobility of MUs, each entry of neighbor table has an aging timer value (*timer*). After receiving a UIM from MU_s, the MU_i needs to update all the neighbor entries of the MU_s by resetting the *timer* to the initial value. When the

Procedure UIM Exchange{}
(1) In a UIM exchange cycle, MU_i constructs a UIM as follows:
For each entry $< id_x, flag >$ in the update list L_{upd}
If $flag == 1$
Get ttl_x value from cached data item d_x;
Else
$ttl_x = 0$;
Calculate the optimal radius (R_{ix}) of d_x;
If $R_{ix} > 0$
$htl_x = R_{ix}$;
Insert $< id_x, htl_x, ttl_x >$ into UIM;
Forward UIM to neighboring nodes;
$L_{upd} = NULL$;
(2) When MU_i receives UIM from MU_s via a neighboring MU:
Reset the *timer* field of MU_s entries in all neighbor tables;
If current UIM is equal to one of previously received UIMs
Discard UIM and return;
For each entry $< id_x, htl_x, ttl_x >$ in UIM
If $ttl_x > 0$
Update the source table and add an entry of MU_s to neighbor table;
If $ttl_x = 0$
delete the entry of MU_s from the neighbor table;
Re-calculate the optimal radius (R_{sx});
If $R_{sx} \leq 0$
delete $< id_x, htl_x, ttl_x >$ entry from the UIM;
If $UIM \neq NULL$
Forward UIM to neighboring nodes;

Figure 4.3 UIM Exchange Algorithm of EPCOR.

timer expires, the entry is deleted from the neighbor table, so that the MU that moved out of the cooperation zone can no longer participate in the cooperative cache sharing. The detailed algorithm of P2P overlay maintenance is shown in Figure 4.3.

Additionally, Figure 4.4 illustrates the proposed EPCOR scheme with a simple example. In this figure, MU_1 is one hop away from MU_2 and MU_3, and two hops away from MU_4. MU_1 caches items d_1,...,d_5. When MU_1 constructs UIM, the *htl* values of these items are 2, 2, 2, 0 and 1 respectively. Thus, a neighbor table entry of MU_1 is

added in the resource tables of MU_2 and MU_3 for the indices of d_1, d_2, d_3, and d_5. MU_1 is also included in the resource table of MU_4 for the indices of d_1, d_2 and d_3. The index of d_5 is deleted from the UIM relayed at MU_2 after the *htl* value of d_5 reaches zero. It is also shown in Figure 4.4 that since the *htl* values of all cached data items in MU_4 are less than 3, the cached data items' information is not included in the resource table of MU_3, which is three hops away from MU_4.

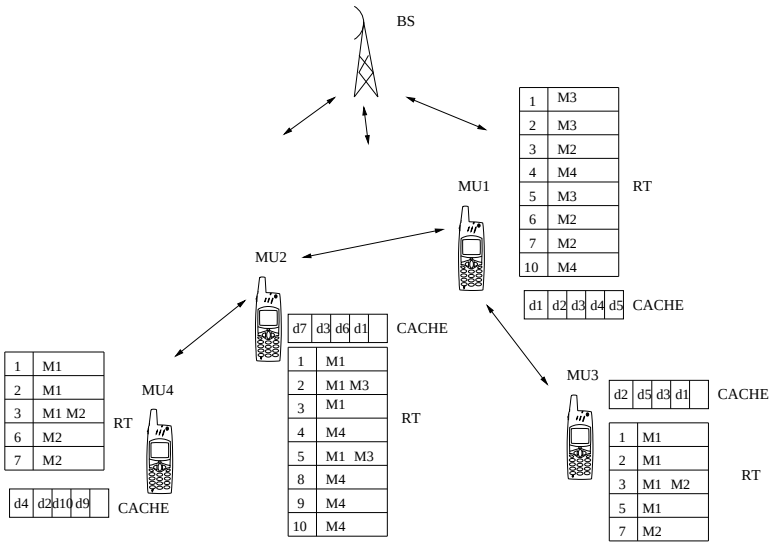

Figure 4.4 A Simple Example of EPCOR Scheme.

4.2.3 Object Lookup and Cache Management

When MU_i initiates a data query, it first checks the local cache for the requested data item. On a cache miss, MU_i searches its resource table. If the requested data item is cached by neighboring peers in the P2P overlay, the query is forwarded to the neighbor with the least hop. Otherwise, the query is forwarded to the BS to retrieve the data item. As shown in Figure 4.4, if MU_1 requests for d_{10}, the request message is forwarded to MU_4. If MU_i receives a query from a source, MU_s, then MU_i first checks

its local cache; if available, the data item is sent to MU_s. If the requested data is not in MU_i's cache, the query is forwarded to the neighboring peer MU_x that cached the requested item and has the least hop from MU_i. If MU_i finds itself as the destination of the query and does not cache the requested item, it means that the source MU_s has inconsistent cache index information of MU_i. In this case, the query is forwarded to the BS to retrieve the data item. For example, in Figure 4.4, MU_4 receives a request message for d_8 from MU_2. In this case, the request message is forwarded to the BS to retrieve the requested data item. Inconsistencies may occur due to loss of UIM or transmission errors. When the source MU receives the requested data item from other MUs or the BS, a cache replacement policy, such as least recently used (LRU), is used to evict the existing data items to make space for the incoming data item. The indices of data items that are evicted from the cache are kept in the update list (L_{upd}) as items with $flag = 0$; and the indices of incoming data items with $flag = 1$ are kept. The update list is used for the construction of next UIM. The algorithm for object lookup and cache management is formally described in Figure 4.5.

4.3 Mathematical Analysis

In this section, we develop an analytical model of the EPCOR scheme to evaluate its performance. An algorithm is developed from the analytical model to calculate optimal radius of the cooperation zone of each data sharing based on energy efficiency.

4.3.1 An Energy Model

In [26] [25],a detailed energy consumption model is proposed for performance analysis of mobile ad hoc networks based on energy consumption measurements of some commercially-available IEEE 802.11 network interface cards (NICs) operating in the ad hoc mode. To the best of our knowledge, this is a most accurate model of energy con-

Procedure Cache Management{}
(1) When MU_i initiates a query for data item d_j
If d_j is in cache /* a cache hit*/
return d_j;
Else If d_j is in the resource table;
choose the neighbor MU_x with the least hop;
send the query to MU_x;
Else
send the query to base station;
(2) When MU_i receives a query;
If requested item d_j is in cache /* a peer cache hit*/
send d_j to source peer;
Else If id_j is in resource table;
choose the neighbor MU_x which has the least hop;
send query to MU_x;
Else If the destination of query is MU_i
send query to base station;
Else
send query to the destination;
(3) When MU_i receives a data item d_j;
If residual cache space is less than the size of d_j;
Evict victim item d_t by using cache replacement policy;
$L_{upd} = L_{upd} \cup < id_t, 0 >$;
cache d_j;
$L_{upd} = L_{upd} \cup \ < id_j, 1 >$;

Figure 4.5 Caching Algorithm of EPCOR Scheme.

sumption in ad hoc networks in the literature. In our analysis, we use this model to calculate the energy consumption of an MU for sending and receiving a message in unicast or broadcast mode. Our analysis of EPCOR scheme is independent of the energy model, other energy models can also be used in our analysis of the EPCOR scheme.

We briefly introduce the energy consumption model presented in [26] and [25]. The energy model and its detailed description about IEEE 802.11 properties are not the major part of this work. Readers can find more details in [26] and [25]. According to

their model, energy consumed by a mobile user using IEEE 802.11 wireless interface for sending, receiving or discarding a message is given by

$$E(s) = m \times s + b \tag{4.1}$$

where s is the message size, m denotes the incremental energy cost associated with a message, and b is the energy cost for the overhead of message. The parameters m and b are different for sending and receiving messages. In particular, the energy model proposed in [26] describes the energy consumption for broadcast and unicast traffic of IEEE 802.11 NICs operating in ad hoc mode.

4.3.1.1 Broadcast Traffic

The IEEE 802.11 standard [104] uses Carrier Sensing Multiple Access with Collision Avoidance (CSMA/CA) technology to avoid frames collision and share channel between multiple senders and receivers. In CSMA/CA, before sending a broadcast packet, the sender listens briefly to the channel. If the channel is clear, the message is sent and received by all nodes in the wireless (radio) range. Otherwise, the sender must back off and try later. In [26], the energy cost associated with sending a broadcast packet is given by:

$$E_{bd\_sd}(s) = m_{bd\_sd} \times s + b_{bd\_sd} \tag{4.2}$$

The cost associated with receiving a broadcast packet is:

$$E_{bd\_rv}(s) = m_{bd\_rv} \times s + b_{bd\_rv} \tag{4.3}$$

4.3.1.2 Unicast Traffic

For unicast traffic, in order to present collisions caused by hidden terminals, a mechanism called virtual carrier sensing is used in addition to CSMA/CA in IEEE 802.11

standard. If a source desires to transmit a message, it first sends a Request-To-Send (RTS) control message identifying the destination. The destination responds with a Clear-To-Send (CTS) message. Upon receiving the CTS, the source sends the data and awaits an ACK from the destination. Thus, the energy cost model for unicast send as shown in [26] is,

$$E_{p2p\_sd}(s) = b_{sendctl} + b_{recvctl} + m_{p2p\_sd} \times s + b_{p2p\_sd} + b_{recvctl} \tag{4.4}$$

where $b_{sendctl}$ and $b_{recvctl}$ are the energy costs for send and receive of small control messages (RTS, CTS and ACK). The cost of a unicast receive is thus given by,

$$E_{p2p\_rv}(s) = b_{recvctl} + b_{sendctl} + m_{p2p\_rv} \times s + b_{p2p\_rv} + b_{sendctl} \tag{4.5}$$

Based on the energy model given in Equations (4.2)-(4.3), we derive the energy cost of broadcasting a message in ad hoc networks. When a packet is broadcast, all nodes within the transmission range of the sender will receive it. The fixed channel access costs are b_{bd_rv} and b_{bd_sd} and the incremental payload costs are m_{bd_sd} and m_{bd_rv}. If r is the radio transmission range, the average number of receivers within the transmission range of the sender is given by $\rho \times \pi \times r^2$, where ρ is the node density of the network. Thus, the total energy cost associated with a broadcast send and receive is,

$$E_{total\_bd}(s) = E_{bd\_sd}(s) + \rho \times \pi \times r^2 \times E_{bd\_rv}(s) \tag{4.6}$$

In ad hoc networks, the energy cost of relaying a message at an intermediate node is given by,

$$E_{relay}(s) = E_{p2p\_rv}(s) + E_{p2p\_sd}(s) \tag{4.7}$$

In IEEE 820.11, messages may be lost due to collision or other failures and the protocol provides for various MAC layer retransmissions. Therefore, each element of protocol in Equations (4.4)-(4.5) includes an implicit factor of $(1 + N_{retransmissions/duplicates})$ where

$N_{retransmissions/duplicates}$ is the average number of retransmissions for sending a packet in the network. Based on this energy model, we analyze the energy cost performance of the EPCOR scheme in the next subsection.

4.3.2 Assumptions and Notations

We introduce the following assumptions made in our analysis.

1. All nodes in the network are two-dimensionally Poisson distributed with density ρ. That is the probability $p(i, A)$ of finding i nodes in an area of size A is given by,

$$p(i, A) = \frac{(\rho A)^i e^{-\rho A}}{i!} \tag{4.8}$$

2. All nodes have the same transmission and receiving range, denoted by r. N is the average number of neighbor nodes within a circular region of radius r. Therefore $N = \rho\pi r^2$.
3. All proxy nodes that exchange data traffic with BS reside within circular area of radius r from the BS.
4. The energy cost of data processing is negligible compared to that due to data communication.
5. All nodes use least recently used (LRU) replacement policy to manage their cache.
6. The data requests of each node follow a Poisson process.

According to [68], the energy cost for transmitting 1K bits of information is approximately the same as the energy consumed to execute 3 million instructions. Therefore, we only consider the energy cost of data communications in our analysis. The other notations used in the analytical model are listed as follows.

- H_i: number of route hops between BS and MU_i;
- H_{ik}: number of route hops between MU_i and MU_k ;
- s_i: size (bytes) of a data item d_i;

- s_{uim}: size of an update index message (UIM);
- s_q: size of a query message;
- CZ_{ij}: the set of MUs in the cooperation zone of MU_i for sharing data item d_j;
- R_{ij}: radius of the cooperation zone CZ_{ij};
- λ_{ij}^q: MU_i's query rate for data item d_j;
- λ_{ij}^{uim}: MU_i's UIM dissemination rate for data item d_j;
- $f_{ij} = \frac{\lambda_{ij}^{uim}}{\lambda_{ij}^q}$: MU_i's UIM dissemination rate relative with query rate for d_j;
- P_{ij}: cache hit ratio of data item d_j in MU_i;
- E_{ij}: overall energy cost of MU_i for accessing d_j;
- ρ: node density of the network.

4.3.3 Problem Formulation

In the EPCOR scheme, each MU maintains a cooperation zone in the localized P2P overlay for each data item by setting the *htl* value in UIM. When MU_i queries for a data item d_j, if d_j is located in the cooperation zone, MU_i will retrieve the data item from the nearest neighbor node, otherwise the query message will be routed to the BS to retrieve the item. The overall energy cost (E_{ij}) of MU_i for accessing d_j can be calculated by considering the energy cost (E_{ij}^1) used to disseminate UIM messages in the P2P overlay and the energy cost (E_{ij}^2) used to retrieve d_j. Thus,

$$E_{ij} = E_{ij}^1 + E_{ij}^2 \tag{4.9}$$

Since the radius of the cooperation zone CZ_{ij} is R_{ij}, the average number of MUs in CZ_{ij} is $\rho\pi R_{ij}^2$. The relative UIM dissemination rate is f_{ij}. In a UIM dissemination process, each MU that belongs to CZ_{ij} will send the UIM once and receive the same UIM

from all neighbors multiple times. Thus, according to the UIM dissemination algorithm (see Fig. 4.3) and Equation (4.6), E_{ij}^1 is calculated as:

$$E_{ij}^1 = \rho\pi R_{ij}^2 \times f_{ij} \times E_{total\_bd}(s_{uim}) \tag{4.10}$$

According to the caching algorithm in Figure 4.5, as MU_i queries for d_j, if d_j is cached in MU_i, we assume the energy cost of data retrieval to be zero. If d_j is cached in some MUs belonging to CZ_{ij}, the item is retrieved from the nearest node. Thus, the average energy cost (e_{ij}^1) for this case is calculated as:

$$e_{ij}^1 = \sum_{l=1}^{R_{ij}} l(1 - \prod_{H_{ti}=l}(1 - P_{tj})) \prod_{H_{ki}<l}(1 - P_{kj}) \times [E_{relay}(s_q) + E_{relay}(s_j)] \tag{4.11}$$

If d_j is not cached in any MU within the cooperation zone CZ_{ij}, the query is forwarded to the BS. The requested item is retrieved from the BS. Thus, the average energy cost (e_{ij}^2) for this case is calculated as follows:

$$e_{ij}^2 = \prod_{H_{ti} \leq R_{ij}}(1 - P_{tj}) \times H_i \times [E_{relay}(s_q) + E_{relay}(s_j)] \tag{4.12}$$

Therefore, the average energy cost (E_{ij}^2) used to retrieve the data item d_j is given as,

$$E_{ij}^2 = e_{ij}^1 + e_{ij}^2 \tag{4.13}$$

The total energy cost of any MU_i to access d_j is thus expressed as:

$$\begin{aligned} E_{ij} &= \rho\pi R_{ij}^2 \times f_{ij} \times E_{total\_bd}(s_{uim}) \\ &+ \left[\sum_{r=1}^{R_{ij}} r(1 - \prod_{H_{ti}=r}(1 - P_{tj})) \prod_{H_{ki}<r}(1 - P_{kj}) + \prod_{MU_t \in CZ_{ij}}(1 - P_{tj})H_i\right] \\ &\times [E_{relay}(s_q) + E_{relay}(s_j)] \end{aligned}$$

In order to optimize the energy consumption of the whole network, we need to minimize the energy cost E_{ij} due to each data access at each node by choosing the optimal

radius (R_{ij}^{opt}) of the cooperation zone CZ_{ij}

Theorem 1. *For any MU_i in the network, there exists an optimal radius $0 \leq R_{ij}^{opt} \leq H_i$ of the cooperation zone CZ_{ij} for sharing any data item d_j, such that the overall energy cost E_{ij} of accessing d_j is minimized.*

Proof. In order to prove the existence of R_{ij}^{opt}, we first calculate the achievable energy saving, when the size of the cooperation zone is increased. Given a radius R_{ij}, we define ΔE_{ij}^1 as the increased energy cost due to UIM dissemination if the radius is increased by one (from R_{ij} to $R_{ij}+1$). Let ΔE_{ij}^2 define the saved energy cost of data retrieval due to larger cooperation zone if the radius is increased by one. According to Equation (4.10), ΔE_{ij}^1 is calculated as

$$\Delta E_{ij}^1 = \rho\pi(2R_{ij}+1) \times f_{ij} \times E_{total\_bd}(s_{uim}) \tag{4.14}$$

and according to Equations (4.11)-(4.13), ΔE_{ij}^2 is calculated as,

$$\Delta E_{ij}^2 = \prod_{H_{ti} \leq R_{ij}} (1-P_{tj})(1 - \prod_{H_{ti}=R_{ij}+1} (1-P_{tj}))(R_{ij}+1-H_i) \times [E_{relay}(s_q) + E_{relay}(s_j)] \tag{4.15}$$

In order to make the problem tractable, we use average cache hit ratio (P_j) of all nodes belonging to CZ_{ij} for data item d_j to calculate the probability of caching d_j in any MU belonging to CZ_{ij}. Thus,

$$P_j = \frac{\sum_{H_{ti} \leq R_{ij}} P_{tj}}{\rho\pi R_{ij}^2} \tag{4.16}$$

According to the Poisson distribution assumption of nodes as in Equation (4.8), the probability that d_j is not cached in any node within a circular area of radius R_{ij} is calculated as,

$$\sum_{i=0}^{\infty} (1-P_j)^i \frac{(\rho\pi R_{ij}^2)^i}{i!} e^{-\rho\pi R_{ij}^2} = e^{-P_j \rho\pi R_{ij}^2} \tag{4.17}$$

Thus,

$$\Delta E_{ij}^2 = e^{-P_j \rho \pi R_{ij}^2}(1 - e^{-P_j \rho \pi (2R_{ij}+1)})(R_{ij} + 1 - H_i) \times [E_{relay}(s_q) + E_{relay}(s_j)] \quad (4.18)$$

$R_{ij} < H_i$ and $\Delta E_{ij}^2(R_{ij}) \leq 0$ implies we can always achieve an energy saving of $|\Delta E_{ij}^2|$ for each data retrieval of d_j, as the radius of the cooperation zone increases by one. Moreover $0 \leq R_{ij} < H_i$, $\Delta E_{ij}^1 > 0$, implies the energy cost of UIM message dissemination increases with the radius of the cooperation zone. If we want to achieve minimal energy cost of each data access of d_j, we need to make sure $\Delta E_{ij}^1 + \Delta E_{ij}^2 < 0$ for each step of UIM dissemination.

The following iterative algorithm finds the optimal radius R_{ij}^{opt}. Initially, R_{ij} is set as 0. At each step, R_{ij} is incremented by 1. In each step, if $\Delta E_{ij}^1 > |\Delta E_{ij}^2|$, the iteration stops and optimal radius, $R_{ij}^{opt} = R_{ij}$. $R_{ij} = H_i - 1$ implies $\Delta E_{ij}^2 = 0$ and $\Delta E_{ij}^1 > 0$, so $\Delta E_{ij}^1 > |\Delta E_{ij}^2|$, thus, the iteration will always stop and find the optimal radius before R_{ij} increases to H_i. Therefore, for any MU_i, the algorithm can find an optimal radius $0 \leq R_{ij}^{opt} \leq H_i$ for the cooperation zone of sharing the data item d_j, so that the energy cost E_{ij} of each retrieval of d_j is minimal. □

4.3.4 Numerical Results

In this subsection, we present some numerical results on the performance of EPCOR based on the model derived above. We evaluate the EPCOR scheme under a hybrid network with one BS and density of $\rho = 100$ nodes/ km^2 and node communication range of $r = 200$ m. The size of a query message is $s_q = 10$ bytes, and the size of UIM is $s_{uim} = 3$ bytes/item. Each MU has the same data access pattern. We consider the energy consumption of MU_i that is $H_i = 10$ hops away from the BS. Also, MU_i accesses a data item (d_j) of size $s_j = 100K$ bytes and average cache hit ratio $P_j = 0.01$.

4.3.4.1 Analysis of an Individual MU

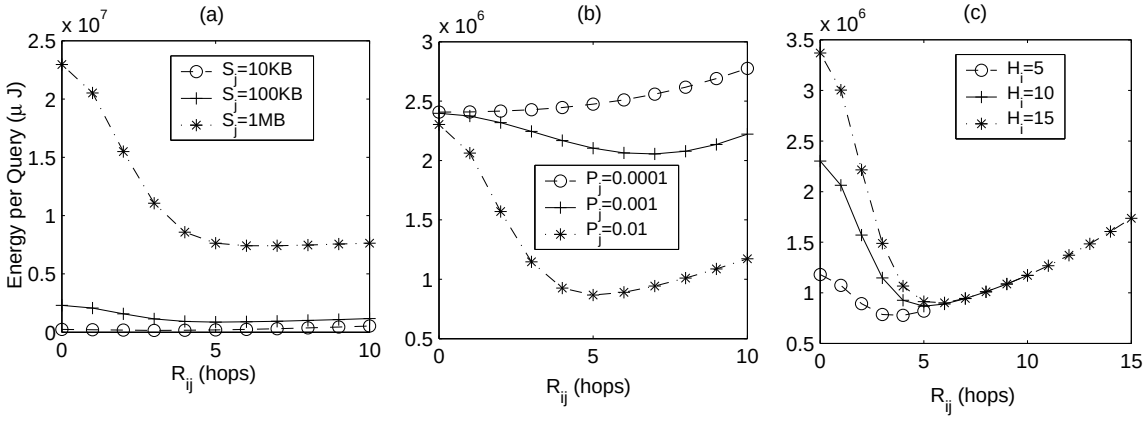

Figure 4.6 Analytical Results for Energy Efficiency Performance of an MU.

First, we investigate the energy consumption of data requests of an individual MU under different data sizes, cache probabilities, and distances from the BS. We assume that indices of 10 items are grouped for one UIM exchange ($f_{ij} = 0.1$). As shown in Figure 4.6(a), there exists an optimal radius for energy cost of data requests for different data sizes. The amount of energy saving for requesting of larger size data items (e.g., $s_j = 1MB$) is much more than that of smaller size data items. This is because the amount of energy cost of data retrieval (E_{ij}^2) is dominant in the overall energy cost (E_{ij}) when the size of the item is large. As the size decreases, we can still save some energy by choosing the optimal radius. If the data size is too small, we set the radius as zero (no sharing with other peers) to prevent the MU from wasting energy.

Figure 4.6(b) shows that the overall energy cost of accessing a data item with high average cache ratio (P_j) is less than that of accessing an item with low average cache ratio. This is because an MU has a higher chance to obtain the popular data items either from its own cache or the neighboring MUs, which reduces the number of hops for a

data retrieval. In order to obtain the data item whose average cache hit ratio is low, the MU needs to search a wider area, thus the optimal radius for a less popular data item is larger than that for more popular data items. As shown in Figure 4.6(b), when the cached probability is 0.0001, the data item does not need to maintain a cooperation zone (i.e., optimal radius equal to zero) for energy efficiency purpose.

Next, we vary the MU's distance (H_i) from the BS to examine its impact on the energy efficiency. As shown in Figure 4.6(c), if there is no cooperation among the MUs (i.e., radius equal to zero), the MU that is 15 hops away from the BS incurs 200% more energy to retrieve the data item than an MU that is 5 hops away from the BS. If we assume that each MU uses optimal radius to form its cooperation zone, the MU with 15 hops needs about 20% more energy to retrieve the data item than the one that is 5 hops away. This feature of EPCOR significantly improves the fairness of energy consumption of each MU at different locations, thus improving the life time of the entire network.

4.3.4.2 Performance Analysis of the Entire Network

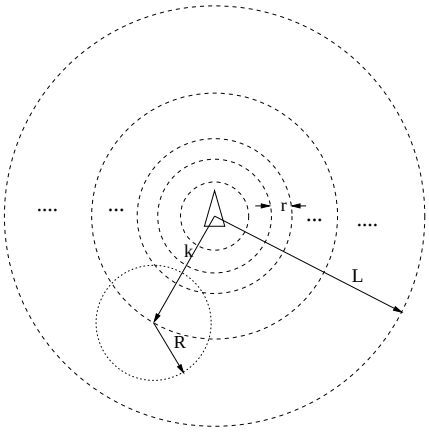

Figure 4.7 A Hyrbrid Wireless Network with Radius L.

As shown in Figure 4.7, we consider a hybrid network in a circular area of radius of L, where the BS resides at the center of the circle. If each MU uses its optimal cooperation radius to form the cooperation zone for each item sharing, we can calculate the optimal energy cost for each data request of MUs having different distances from the BS by using the iteration algorithm proposed in proof of Theorem 1. Let $E_k(R_{ij}^{opt})$ denote average energy cost for each data request of MUs that are k hops away from the BS. In the network, the average number of MUs that are k hops away from the BS is equal to $\rho\pi(k^2-(k-1)^2)$ where ρ is the node density. The total average number of MUs in the network is $\rho\pi L^2$.

We use the average *energy per query* (EPQ) as a metric to evaluate the energy consumption of each query in the whole network. EPQ is defined as the sum of energy consumptions of all MUs divided by the total number of queries in a given period. Thus, in the time period t, each MU has $\lambda \times t$ data requests, where λ is the average query rate from each MU. There are $L^2\pi\rho\lambda t$ data requests generated in the whole network. Therefore, the EPQ performance of EPCOR for the whole network is given as:

$$EPQ = \frac{\sum_{k=1}^{L}(k^2-(k-1)^2)\pi\rho E_k(R_{ij}^{opt})\lambda_j t}{L^2\pi\rho\lambda t} = \frac{\sum_{k=1}^{L}(2k-1)E_k(R_{ij}^{opt})}{L^2} \tag{4.19}$$

Since the wireless channel bandwidth of BS is the bottleneck in determining the throughput of hybrid wireless networks, we use another metric, called *base station retrieval ratio* (BRR) to evaluate the burden of the base station. BRR is defined as the number of data items retrieved from the BS divided by the total number of requested data items of all MUs in the network. If BRR is high, it means that the channel is more congested since more data items are retrieved from the BS. According to Equation (4.17), the probability of a cache miss in the cooperation zone of the MU that is k hops from the BS is $e^{-P_i\pi\rho(R_{ij}^{opt})^2}$. Therefore, the BRR of the whole network is given by:

$$BRR = \frac{\sum_{k=1}^{L}(k^2-(k-1)^2)\pi\rho e^{-P_i\pi\rho(R_{ij}^{opt})^2}\lambda t}{L^2\pi\rho\lambda t} = \frac{\sum_{k=1}^{L}(2k-1)e^{-P_i\pi\rho(R_{ij}^{opt})^2}}{L^2} \quad (4.20)$$

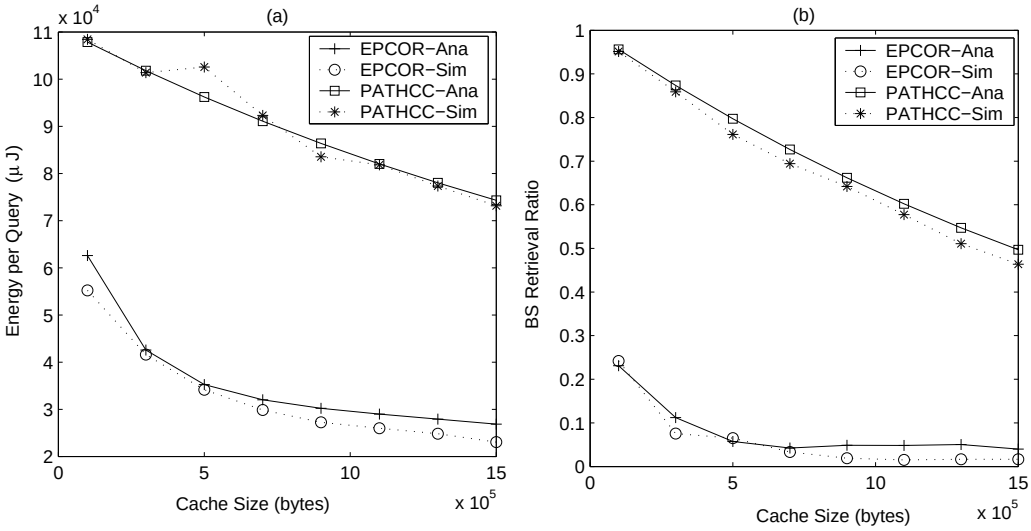

Figure 4.8 Analytical and Simulation Results of a Whole Network.

Now we present the analytical results of the EPQ and BRR performances for a simple hybrid wireless network. The analytical (Ana) results are also compared with the simulation (Sim) results. We consider a hybrid wireless network in a circular area of radius $L = 1200$ meters. Each MU has the same cache space and transmission range of $r = 200$ meters. There are 1000 data items in the system, each of size 10 Kbytes and same popularity for all MUs. The size of UIM is $s_{uim} = 2$ bytes/item. In this evaluation, we compare EPCOR with path cooperative caching (PATHCC) [102], in which the cooperation radius is set to zero and only the nodes that reside in the route path from the source node to the BS participate in the cooperation to share the cache data. In our simulations, all MUs are assumed to be stationary and the packet delivery to nodes is instantaneous and error-free. This simulation thus faithfully represents the request generation and data retrieval for the different caching schemes.

As shown in Figure 4.8(a), the EPQ performance of both EPCOR and PATHCC drop as the cache size increases. A network with EPCOR deployed saves almost 70% energy compared to that using PATHCC. Figure 4.8(b) shows that EPCOR has much lower BRR than PATHCC, thus significantly reducing the traffic at the base station. Both Figures 4.8(a) and (b) demonstrate that the simulation results match the analytical results very closely, thus validating our system model.

4.4 Implementation Issues

In this subsection, we address four critical implementation issues, namely estimation of run-time parameters, incremental calculation of optimal radius, maintenance of data consistency and piggyback UIM message dissemination. Additionally, the time and space complexity of EPCOR are also discussed.

4.4.1 Estimation of Run-time Parameters

As shown in Equations (4.14) and (4.15), several parameters are involved in calculating the optimal radius. The incremental energy coefficient (m) and the overhead (b) in Equation (4.1) are system parameters obtained from the specifications of the wireless network card. The UIM dissemination frequency (λ_{ij}^{uim}), size of UIM (s_{uim}) and size of request message (s_q) are obtained from the settings of EPCOR. Data query rate (λ_{ij}^{q}) can be estimated by using the sliding average method [82]. The distance H_i of each MU in terms of the number of hops from the BS is available at the routing table. The node density (ρ) of the whole network can be estimated at the BS and disseminated to all MUs.

In order to estimate the cache hit ratio of data items, we assign a metadata for each requested data item d_j (including local and peer requests) in the MU_i. The metadata includes two counters: query counter (c_j^q) and hit counter (c_j^h). When MU_i initializes

a query for data item d_j, or MU_i receives a query from other peers, the corresponding request counter c_j^q is incremented by 1. When MU_i satisfies a query for d_j, the hit counter c_j^h is incremented by 1. Thus, the hit ratio of the data item d_j at MU_i is estimated as $P_{ij} = c_j^q / c_j^h$.

4.4.2 Incremental Calculation of Optimal Radius

In Equation (4.18), we use average cache hit ratio (P_j) to calculate the optimal radius. MUs have different cache hit ratios for the same data item and the hit ratio of each item may dynamically change in an MU. Therefore, in order to efficiently estimate P_j dynamically and calculate the optimal radius R_{ij}, we use an incremental method.

In this method, P_j is added as an additional attribute to the entry of UIM for d_j. When MU creates a UIM message, its local hit ratio of d_j is set as the value P_j. In each broadcast step, if MU_i receives a UIM from a neighboring MU, this attribute is updated as $P_j = \frac{P_j * n + P_{ij}}{n+1}$, where n is the number of MUs the UIM has visited after leaving from the source MU and P_{ij} is the hit ratio of d_j at MU_i. After MU_i receives the UIM, ΔE_{ij}^1 and ΔE_{ij}^2 are calculated according to Equations (4.14) and (4.18). According to the proof of Theorem 1, if $\Delta E_{ij}^1 > |\Delta E_{ij}^2|$, MU_i stops dissemination of the UIM message. Otherwise, MU_i broadcasts the UIM message to all neighbor nodes. Therefore, the average cache hit ratio P_j and optimal radius are incrementally calculated as UIM propagates through the cooperation zone in each dissemination step. Since this method does not require global cache hit ratio information of all MUs in the cooperation zone, the method can significantly reduce the computation cost of the calculation of optimal radius.

4.4.3 Data Consistency

In this chapter, we assume that all data updating occurs at the BS. The data item is associated with a time-to-live (TTL) value after retrieval from the BS. When an MU

receives a query for a cached item d_j whose TTL value has not expired, the MU responds to the query with the valid data item. If the requested data item is stale, the request is forwarded to the BS or other peers to retrieve the valid one. In estimating the hit ratio, we consider a stale hit the same as a cache miss, so a data item with low TTL value (i.e., invalidated frequently) has a high cache miss ratio.

4.4.4 Piggyback UIM Message Dissemination

In most ad-hoc routing protocols (e.g., AODV [66], DSDV [65], or ZRP[39]), two neighboring MUs need to frequently exchange hello messages in the routing layers to discover and maintain their neighborhood relationship. Due to the channel contention [26], the energy cost of independently sending a small size message is much more than the energy cost of piggybacking with other messages. This feature motivates us to use a piggyback dissemination mechanism to propagate UIM messages of the EPCOR scheme through the P2P overlay network. In the piggyback dissemination scheme, after a UIM message is created as shown in Figure 4.3, a pending message queue is used to store the created UIM messages. When the routing layer notifies a hello message exchange, the UIM messages in the pending queue are piggybacked with hello message, and broadcast to all neighboring nodes. The neighboring nodes can extract UIM message from the received hello message and update their resource table and neighbor table. If the UIM dissemination is not stopped, the UIM message will be inserted into pending queue for subsequent broadcast. Otherwise, the UIM message is deleted from the pending queue. Since the piggyback dissemination does not require each MU to proactively send UIM messages, it significantly reduces the energy and bandwidth requirement of UIM message dissemination. Furthermore, it also eliminates the chance that nodes cannot go to the sleep mode due to UIM message dissemination, hence it improves the energy saving at each node.

4.4.5 Complexity

From Figure 4.3, the time complexity to construct and process a UIM is $O(n)$, where n is length of L_{upd}. Each MU needs $O(D * M)$ memory space to maintain a resource table and all neighbor tables, where D is the maximum number of entries of a resource table and M is the maximin number of MUs in a neighbor table. From Figure 4.5, a lookup operation in resource table requires $O(log\ D + log\ M)$ time, if we use priority queues to maintain the resource and neighbor entries.

4.5 Performance Evaluation

In this section, we evaluate the performance of the EPCOR scheme through simulation experiments. Different metrics are used to evaluate four main aspects of its performance. They are energy efficiency, access latency, throughput and load balance.

4.5.1 Simulation Model and System Parameters

We use the Network Simulator (*NS-2*) [108] to study the performance of EPCOR. Since EPCOR does not rely on any specific routing protocol, in our simulations, we use DSDV [65] as the routing protocol to route data traffic in the hybrid wireless network. Our simulations use IEEE 802.11 as the medium access control (MAC) protocol and also use two-ray ground reflection as the radio propagation model. The default transmission range of each MU is $r = 200$ meters. For the mobility model, as shown in Figure 4.9, we assume a service area of 1500m $\times$ 1500m. One fixed base station is located at the coordinate (750m, 0m). All mobile nodes moving in this area follow the random way point mobility model [11], which is commonly used to model the movement of individual pedestrians. According to this model, each mobile node starts at a location chosen uniformly at random inside the service area. For each movement, the target location is also randomly chosen, and the moving speed is uniformly random in the range $[0, v_{max}]$.

After the mobile node reaches its destination, it pauses for a period of time (t_p) before continuing its next movement.

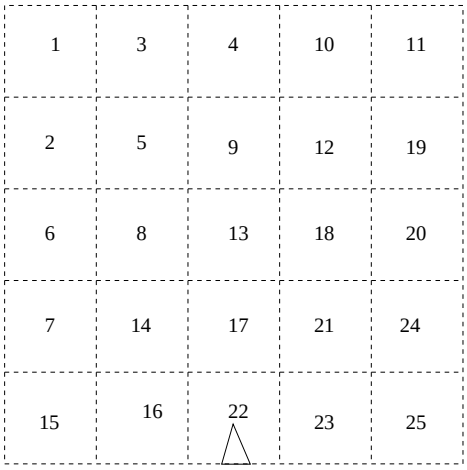

Figure 4.9 A Simulated Hybrid Network.

For Web data, the data size distribution follows the lognormal model [7]. In our simulation also, the lognormal model is used for the size distribution of all data items in the system. The cutoff minimal size of the data item is assumed to be 128 bytes, the cutoff maximal size is 40 Kbytes, and the average size is 12.5 Kbytes. The update intervals (i.e., time-to-live values) of all data items follow uniform distribution in the range of [0, ttl_{max}]. The data requests of each mobile user follow a Poisson process. After a request is sent out, if the MU does not receive the data item, it waits for an interval (t_w) before resending the same request message. The MU does not generate new request until the query is served. We consider *Zipf-like* distribution [10] for the data popularity pattern of MU's access, in which the access probability a_i of data item d_i is proportional to its popularity rank, *rank(i)* where the most popular item has the smallest rank value. That is, $a_i = A/rank(i)^{\theta}$, where A is the normalization constant and θ is a parameter between 0 and 1. As shown in Figure 4.9, the entire service area is equally divided into

5×5 square grids. Beginning from the left upper most grid, we index the grids as 1, 2, 3,..., 25 in a zig-zag diagonal-wise fashion. The MUs in the same grid have the same *Zipf* based data popularity pattern. MUs in different grids have different shift values for the *Zipf* pattern. For an MU in grid i, the *id* of data access is shifted by i so that $id = (id+i)$ mod N, where N is the total number of data items in the system. According to this model, the data access of two adjacent grids show high similarity while the data access of two far-off grids show a low similarity. In the simulations, a first come and first serve (FCFS) policy is used at the BS to serve the incoming data requests. The settings of system parameters are given in Table 4.1. Some parameters may vary in the following experiments.

Table 4.1 System Parameter Setting

Parameters	Default values	Range of values
Number of mobile devices	120	NA
Number of data items	3000	NA
Cache size	1500 $Kbytes$	500-2500 $Kbytes$
Mean access rate (λ_i)	1/30 sec^{-1}	1/10-1/40 sec^{-1}
Average Update Interval	10000 sec	625-10000 sec
Pause time (t_p)	2 sec	NA
Max moving speed (v_{max})	2 m/s	2-20 m/s
Initial value of aging timer ($timer$)	120 sec	NA
Bandwidth	11 $Mbps$	NA
UIM exchange Interval (T)	60 sec	NA
Waiting interval (t_w)	5 sec	NA
Request message size (s_q)	10 $bytes$	NA
UIM message size (s_{uim})	3 $bytes/item$	NA

4.5.2 Simulation Results

For each experiment, we simulate for 5000 seconds and the results after initial 1000 seconds are recorded for performance evaluation. All MUs' caches are filled with

randomly chosen data items at the initial state. Four major metrics are used in the evaluation: *hit ratio* (HR), *peer hit ratio* (PHR), *energy per query* (EPQ) and *average latency*. We define PHR as the ratio of the number of data items retrieved from peer mobile users to the total number of requested data items of a mobile user. Recall that EPQ is the ratio of the total energy consumed in a given period to the total number of requests in that period. Under the same data request pattern, EPQ reflects the energy consumption performance of different schemes. Energy consumption of each mobile device is measured by the linear model proposed in [26].

In the performance study, two variants of the EPCOR scheme are evaluated. In the basic EPCOR scheme, each MU disseminates UIM messages independently. In the enhanced piggyback scheme (EPCOR-PG), MUs piggyback UIM messages with the underlaying routing hello messages. These two variants of EPCOR scheme are compared with three other schemes, namely NOCACHE, PATHCC and EXPRING. In the NOCACHE scheme, no cache space is used for each MU so that every data item is retrieved from the base station. In path cooperative caching (PATHCC) scheme [102], each node does not have resource table to maintain the neighbors information so that each data request is forwarded to the BS, and only the nodes on the route path from the source node to the BS participate in the cooperation to share the cached data. A broadcast search based cooperative caching scheme has been proposed in [53] for the hybrid wireless networks. In our comparison, in order to reduce data traffic of request message flooding, the expanding ring (EXPRING) scheme [60] is deployed to set the range of request message flooding. This algorithm can be viewed as successive instantiation of flooding search with increase in TTL value ranging from 1 to the number of hops from the BS. In the default setting, the waiting time between two consecutive broadcasts is 5 seconds. In our simulations, the same data access patten and mobility model are applied to all the above schemes. The same least recently used (LRU) cache replacement policy is also

deployed in PATHCC, EXPRING and EPCOR schemes.

(1) Effect of Cache Size

In this experiment, we evaluate the performance of EPCOR under different cache sizes and compare with the other three schemes. In Figure 4.10, the lower part of each column represents the hit ratio (HR) of the corresponding scheme, and the upper part of each column represents the peer hit ratio (PHR) of the scheme. We observe that for identical cache sizes, four schemes show almost the same HR performance, since all four schemes use the LRU cache replacement policy. The EXPRING and EPCOR-PG schemes have the best PHR performance (note that EXPRING uses flooding to locate the nearest requested data item in the network). The EPCOR outperforms the PATHCC in terms of PHR, and NOCACHE have zero HR and PHR performance since no cache is used in each MU. The enhanced EPCOR-PG significantly improves PHR performance over basic EPCOR, since the former uses piggyback method to disseminate UIM messages which reduces the cost of UIM dissemination so that each MU can maintain a larger cooperation zone for each data item. Figure 4.10 also shows that the HR and PHR performance of EXPRING, EPCOR and PATHCC schemes increase with larger cache sizes.

Figure 4.11 shows that EPCOR performs better in terms of EPQ than the other three schemes. This is because EPCOR maintains a P2P overlay to share the cached data, so that the number of hops for one data item retrieval is minimized. Moreover, the energy cost of UIM dissemination is limited by using optimal radius value. Although EXPRING can locate the nearest requested data item, the energy cost of the flooding search is huge for a wireless network, so EXPRING has the worst EPQ performance. Because PATHCC deploys a cache at each MU and the peers between the source node and BS participate in the cache sharing, PATHCC consumes less energy for each data retrieval than NOCACHE which does not deploy cache for each peer. In Figure 4.11, the

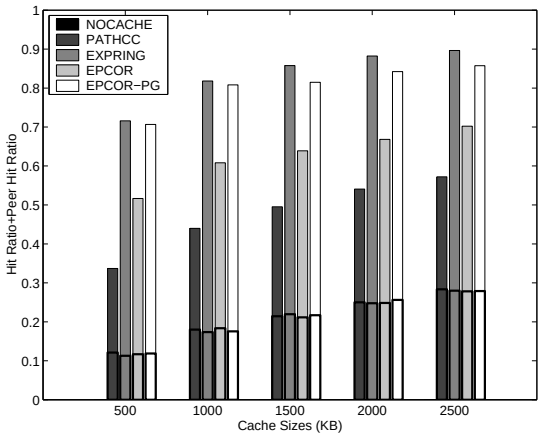

Figure 4.10 Hit Ratio (HR) and Peer Hit Ratio (PHR) vs Cache Size.

basic EPCOR scheme consumes about 40% less energy than PATHCC. The enhanced EPCOR-PG scheme has better EPQ performance than the basic one. This is because of energy saving in UIM piggyback dissemination and better PHR performance.

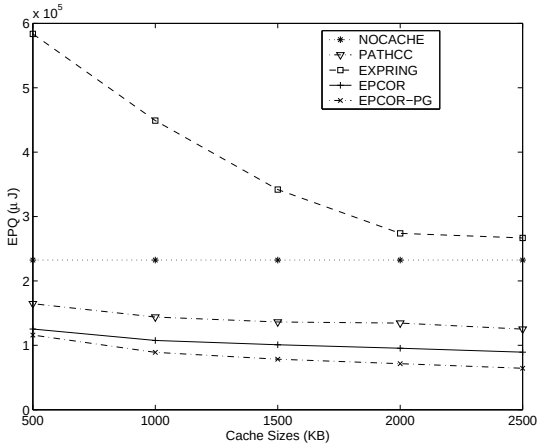

Figure 4.11 Energy per Query (EPQ) vs Cache Size.

Figure 4.12 shows that the EPCOR schemes outperform the other three schemes in terms of average access latency. EXPRING, EPCOR and PATHCC achieve a performance improvement with increasing cache size, while the improvement of EPCOR and

EXPRING is greater than that of PATHCC. This is because only few MUs participate in the cooperation for data sharing in PATHCC, as the probability of retrieving the requested data item from the nodes on the path toward BS improves slightly when the cache size increases. As shown in Figure 4.12, since both PATHCC and EPCOR have opportunities to retrieve the data from other nearby peers, they have low access latency than NOCACHE.

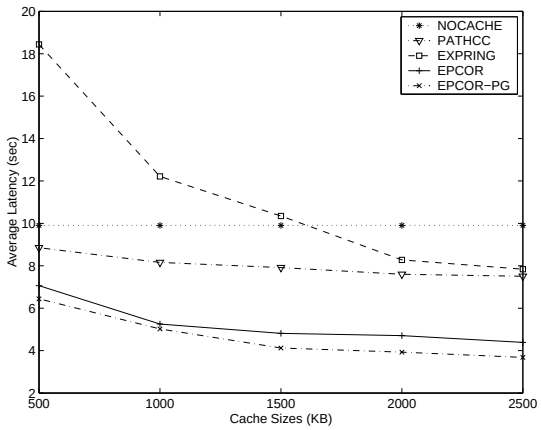

Figure 4.12 Average Access Latency vs Cache Size.

(2) Effect of Query Rate

In this simulation, we change the average query interval of each peer to investigate the scalability of each scheme. The cumulative data traffic of base station in 4000 seconds is measured as the performance metric. Figure 4.13 shows cumulative BS data traffic during 4000 seconds simulation under different query rates. In the experiment, we use two different waiting intervals (5 seconds and 10 seconds) for EXPRING flooding scheme to evaluate its effect on the performance of bandwidth consumption and network throughput. EXPRING-5 and EXPRING-10 represent the scheme with 5 and 10 seconds waiting intervals, respectively.

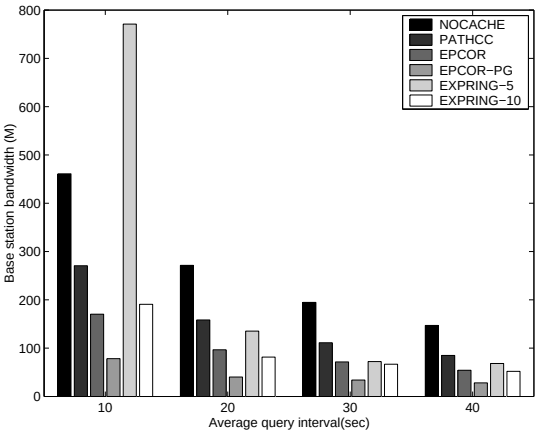

Figure 4.13 BS Bandwidth Consumption vs Query Rat.

As shown in Figure 4.13, NOCACHE has high BS traffic at low and moderate query rates (i.e., 1/40-1/20 sec^{-1}). This is because all requested data items are retrieved from the BS in this scheme. EXPRING and EPCOR have similar BS data traffic under low and moderate query rates. They retrieve the most requested items from other peers so that the traffic is distributed in the whole network, which relieves the traffic burden on the BS. However, when the query rate is high (i.e., 1/10 sec^{-1}), the BS data traffic of EXPRING-5 dramatically increases. This is because EXPRING-5 needs to flood its request message every 5 seconds if the requested data item can not be located. Under high query rate, the flooding messages quickly exhaust the network bandwidth. Most nodes fail to retrieve the items from other peers, so that the nodes have to increase the flooding range until they reach the base station. EXPRING-10 reduces the flooding traffic by using longer waiting interval (10 seconds), which reduces the traffic congestion at BS. The P2P overlay of EPCOR enhances peer to peer data retrieval among all nodes, and the dynamic cooperation radius setting reduces the UIM dissemination traffic (e.g., most nodes around the BS do not generate UIM traffic). Thus, EPCOR has the least BS bandwidth consumption under different query rates. If we consider the bandwidth of BS as the bottleneck of the throughput of the entire network, EPCOR improves the network

throughput by efficiently enhancing the peer-to-peer data retrieval among all nodes in the network. In the enhanced EPCOR scheme, since piggyback mechanism is used to disseminate the UIM messages, the network traffic is significantly reduced. Figure 4.14 shows the access throughput of six schemes under different query rates. We measure the total number of successful data queries in the given simulation period to indicate the throughput performance of these schemes. As shown in the figure, EPCOR has the best throughput among all schemes under different query rates. Due to the network congestion, the expanding ring scheme suffers a low throughput at the high query rate.

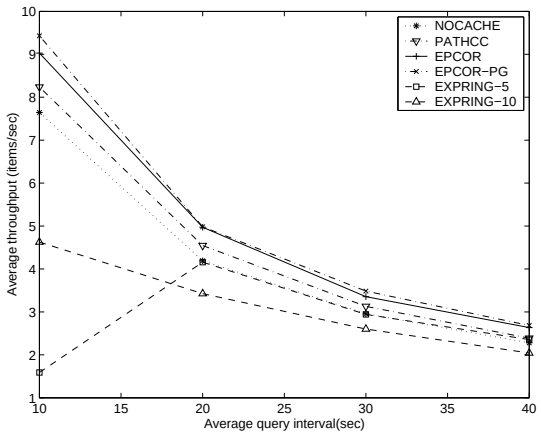

Figure 4.14 Average Throughput vs Query Rate.

(3) Effect of Data Update Rate

In this experiment, we evaluate the performance of EPCOR under different data update rates. Figure 4.15 shows the performance of HR and PHR of five schemes. As the update rate increases, both HR and PHR metrics decrease. This is because fewer and fewer valid data items are cached in each node to serve local queries and those from other peers.

Figures 4.16 and 4.17 show the access latency and EPQ performances of four schemes under different data update rates. Figure 4.16 shows that the access laten-

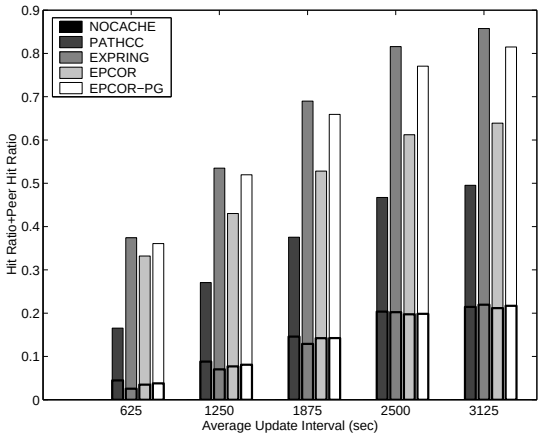

Figure 4.15 Hit Ratio and Peer Hit Ratio vs Data Update Rate.

cies of EXPRING, PATHCC and EPCOR increase as the data update rate increases. PATHCC and EPCOR have similar access latency as NOCACHE at high data update rate, and EXPRING suffers long access latency at high data update rates. When data items are updated frequently, few valid items are available at the cache of each node and therefore most queries are forwarded to the BS to retrieve the data items. In the EXPRING scheme, if one query fails, the node needs to increase the broadcast hops for the next query. Therefore, when update rates are high, most queries need to reach BS to retrieve the requested item, leading to a broadcast storm in the network. Thus, EXPRING has a long access latency and high energy consumption at high data update rates.

(4) Load Balance and Effect of Mobility

In order to investigate the load balance in each scheme, we run the simulation for 4000 seconds and record the energy consumption of each MU under three different mobility scenarios: $v_{max} = 2m/s$, $10m/s$ and $20m/s$. Because the energy cost of each MU in the simulation period indicates the amount of data traffic that the MU has processed, we calculate the standard deviation of energy consumption of all MUs at the end of the experiment to measure the load balance of each scheme. Figure 4.18 shows the energy

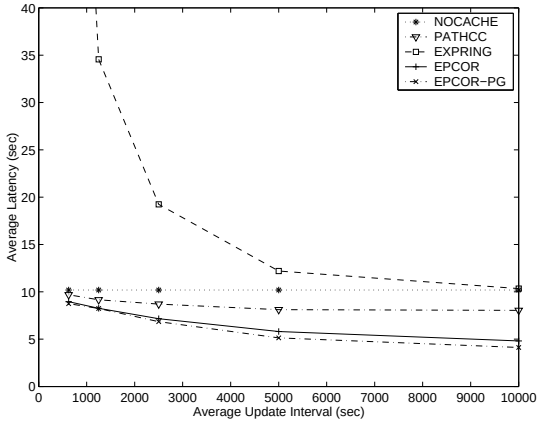

Figure 4.16 Average Access Latency vs Data Update Rate.

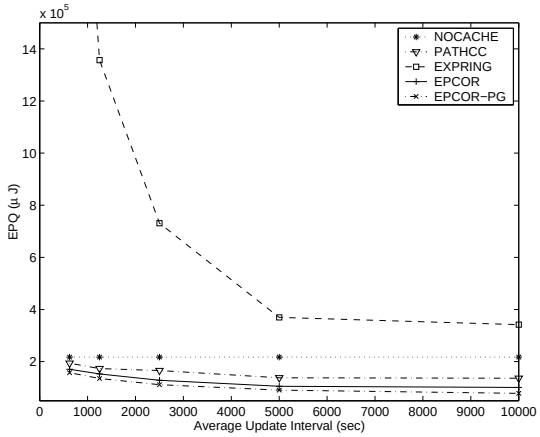

Figure 4.17 Energy per Query vs Data Update Rate.

consumption standard deviation (ECSD) of all MUs for each scheme after 4000 seconds of simulation.

It is shown in the figure that each scheme has a high ECSD performance at low mobility scenario (i.e., $v_{max} = 2m/s$), and ECSD drops as the mobility increases. Under a low mobility scenario, the MUs that are close to the BS need to relay data traffic between the BS and other MUs. Therefore, the MUs around the BS have a much higher energy consumption than the MUs that are far away from the BS. This results in a poor load balance in hybrid networks. As shown in Figure 4.18, NOCACHE scheme

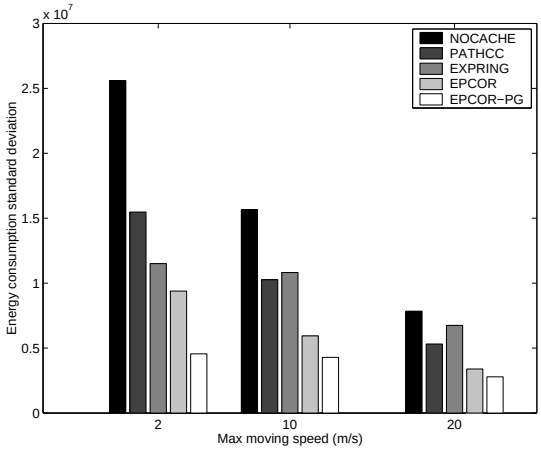

Figure 4.18 Energy Consumption Standard Deviation vs Mobility.

has a high ECSD at low mobility. This is because no cache is deployed at each MU in NOCACHE and all queries are forwarded to the BS to retrieve data items. Thus the MUs closer to the BS consume more energy due to the frequent data relaying, leading to high ECSD performance. Since the battery life of the MUs that relay data traffic to BS is the indicator of the life time of the entire network, NOCACHE has a short life time due to the poor load balance. Figure 4.18 shows that EPCOR outperforms the other three schemes under different mobilities. This is because the P2P overlay of EPCOR uses a cooperation zone to share each data item among all peers, and the radius of cooperation zone varies according to the distance from BS. The data items of far-off MUs have a bigger cooperation radius than that of nearby MU. Thus, the data traffic is equally distributed in the whole network, and EPCOR significantly improves the load balance performance of hybrid networks. Enhanced EPCOR maintains bigger cooperation zone of each data item due to the low cost of UIM piggyback dissemination, hence it improves load balance among all nodes in the network. As the mobility increases, ECSD of each scheme drops. In a high mobility scenario, the average moving speed of each MU is high so that each MU has equal opportunity to appear at different locations of the hybrid networks. Thus, the

energy consumption of nodes is similar to one another as the network topology changes fast.

4.6 Summary

Inheriting positive features of infrastructure-based and mobile ad hoc networks, the hybrid wireless networks are expected to become effective and popular in providing ubiquitous information services. In such networks, mobile users can pull information via cellular networks as well as retrieve requested information from other mobile users via multi-hop peer-to-peer wireless communications. In this chapter, we proposed a novel caching scheme, called *energy efficient peer-to-peer caching with optimal radius* (EPCOR) to reduce latency and energy consumption associated with data accesses for each mobile user in such networks. In this scheme, a peer-to-peer (P2P) overlay is built based on network proximity and data preference to enhance peer-to-peer communication and load balancing among all mobile users. In the P2P overlay, each MU shares a data item in a cooperation zone by proactive dissemination of cache index information.

The cooperative information sharing in the EPCOR scheme is only limited in all mobile users in the same cooperation zone. If the requested data is cached by other mobile users outside of the cooperation zone, the data query needs to be forwarded to the base station and pull the information from the original sever via cellular networks. However, in some application scenarios, like moving vehicles,subway stations, or battle field, there is no wireless infrastructure available. How to enable ubiquitous information services to mobile users in such scenarios is a big challenge to our proposed framework. Moreover, since more and more mobile devices equipped with Global Position System (GPS), and location information is available to mobile users. How to utilize the location information of mobile users to improve the performance of information access in the mobile and distributed environments is a another challenge for our proposed framework. In the

next chapter, we will investigate location-aided data retrieval in the mobile environments without wireless infrastructure.

CHAPTER 5
LOCATION-AIDED DATA RETRIEVAL IN MOBILE P2P NETWORKS

In the vision of next generation of mobile systems, mobile ad hoc networks (MANET) [107] will be used to provide information access to mobile users in the environments without infrastructure networks. With more mobile devices equipped with Global Positioning System (GPS), location information of mobile users can be utilized to improve the performance of information access in mobile ad hoc networks. In this chapter, we propose a novel scheme called *Proximity Regions for Caching in Cooperative MP2P Networks* (PReCinCt). PReCinCt scheme utilizes the location information of each mobile user to improve the performance of peer-to-peer data retrieval in large scale mobile ad hoc networks.

Along with the rapid advancements in Peer-to-Peer (P2P) computing, mobile peer-to-peer (MP2P) networks [48] are being introduced to help mobile users share their data via wireless peer-to-peer communications. The MP2P system comprises P2P overlay networks on top of mobile ad hoc networks. In contrast to P2P systems on wired networks that comprise static peers, MP2P systems are subjected to the limitations of battery energy, wireless bandwidth, and the highly dynamic nature of the network topology. In P2P networks, data retrieval scheme is used by each peer to locate and retrieve requested data items from other peers. In existing unstructured P2P networks [105], flooding is the most popular data retrieval mechanism. Flooding entails message processing at every node which is expensive in terms of bandwidth, battery power and computational resources. In [60], the expanding ring scheme is proposed to reduce the cost of each data retrieval. A node starts a request flooding with a small Time-to-Live (TTL) and continuously increases the TTL until the data is found. Due to the resource constraints

(e.g., battery and wireless bandwidth), both flooding and expanding ring schemes do not adapt well in large scale MP2P networks. In the structured P2P networks [85], distributed hash table (DHT) is used to deploy and retrieve data in controlled network topologies. However, due to the users' frequent mobility, the data retrieval schemes of structured P2P networks do not perform well in MP2P networks.

In PReCinCt scheme, the entire network topology is divided into geographical regions, each being responsible for a set of keys representing the data of interest. A geographical hash function $h(k_i) = L_j$ is used at each peer to map a key k_i to a region (R_j)'s location L_j. We call the region R_j as the *home region* of key k_i. When a peer requests a data item represented by a key k_i, the peer obtains the location information of home region R_j by using the hash function, and then a request message is sent to the home region R_j by using a geographic-aided routing protocol, such as GPSR [33]. After reaching the home region, localized flooding is used to locate the peer holding the requested data. By routing to regions rather than to specific points, PReCinCt requires only approximate location information of each region, thus making it robust to errors in location measurement and frequent mobility of peers. Moreover, a cooperative caching scheme is integrated with PReCinCt scheme to further improve data retrieval performance of mobile users. In addition, we also propose a hybrid push/pull algorithm called *Push with Adaptive Pull* that uses minimal message overheads to maintain data consistency for PReCinCt scheme.

5.1 Description of PReCinCt Scheme

This section describes our PReCinCt scheme for data retrieval in MP2P networks. In this scheme, the network topology is divided into geographical regions, each associated with a set of keys. The keys for the data are distributed among the regions, such that each key k_i maps to a region R_j. Peers can determine the key-set to region mapping

by using a geographic hash function. Every data item is associated with a *home region*, where the data item is initially stored.

5.1.1 Region Management

In PReCinCt, the whole network topology is divided into multiple geographical regions. Each region is represented by the location information of its center point and all vertices in the perimeter. Each peer uses a region table to keep the location information of all regions in the whole network. When a peer joins the network for the first time, the peer can retrieve the region table from its neighboring peers. The size and shape of the regions can be changed by updating the region table of all peers in the network, at the same time each key in the network also need to be relocated according to the region table changes.

There are four operations that can be applied to regions: *Add*, *Delete*, *Merge*, and *Separate*. In add operation, a new entry which includes the location information of a new region is added into the region table to indicate the expansion of the whole network topology. When a region no longer belongs to the network, the delete operation will remove the entry of corresponding region from the region table. In merge operation, the location information entry of the new region replaces the entries of two existing region in the region table to indicate the merging of two neighboring regions into one. Separate operation is used to divide one region into two new regions. All of these four operations are executed by updating the region location information in the region table. After each execution, the peers need to disseminate the update to all other peers in the whole network to guarantee the consistency of region tables of all peers.

5.1.2 Data Search Process

When a peer requests for a data item, it first floods the request in the region in which it currently resides to determine if any of its neighboring peers have a local cached

copy of the item. If so, it is a local hit, the request succeeds and the response is sent back to the requesting peer. If not, the requesting peer uses the geographic hash function to get a location according to the query. Home region is determined by searching the region whose center location is closest to the location from the region table. The requesting peer generates a request message containing the following three fields: i) the identity of the peer making the request, ii) the location of home (destination) region for the requested item, and iii) the key of data item being requested.

Nodes routing the request message towards the destination region, check the destination region location in the header to determine whether they are within that region. The first node inside the destination region receiving the request message is identified as the *point of broadcast*. It floods the message within the region to locate the peer holding the data item. Each peer in the home region processes the request message to determine if it has the requested data item. Peers located outside the home region drop the request message without further processing. When the data item is located, the response is sent back to the original requesting peer and the search process terminates. When the requesting peer receives the response, the cache manager decides if this data item should be cached. Thus using the PReCinCt scheme, flooding is limited to within a region which leads to savings in network bandwidth and energy consumption of the nodes. The detailed algorithm for search process in PReCinCt is given in Figure 5.1.

5.1.3 Peer Mobility Handling

In PReCinCt, mobility of peers is classified into two categories: *intra-region* and *inter-region*. In the intra-region mobility, a peer moves in the same region which causes minimal overhead in PReCinCt. In the inter-region mobility, on the other hand, a peer moves out of its initial region to a neighboring region. Each peer checks its position periodically to detect a inter-region mobility. If inter-region mobility occurs, the peer

d_j: requested data item;
P_j^q: peer requesting the d_j;
R_j^q: region in which P_q resides;
P_j^p: peer which responds with data d_j ;
R_j^h: home region of data item d_j;

Procedure Search (query of d_j)
Begin
if (d_i is cached in P_j^q)
Update utility value and return d_j;
else
Localized broadcast the query in R_j^q
for (each peer P_i in R_j^q)
if (d_j cached in some peer P_j^p)
Update utility value of d_j;
Send d_j to P_j^q
else
Get the location of home region R_j^h by using GHT;
Route the query to R_j^h
Localized broadcast the query in R_j^h
Reply the query with d_j to P_j^q
End

Figure 5.1 The Search Algorithm of PReCinCt Scheme.

has to distribute its keys to other peers in the original region. To reduce the overhead of inter-region mobility, the peer moving out of a region sends its keys to the other peers that satisfy the following criteria: 1) have low mobility rates; 2) are located near the center of the region; and 3) have cache space to store this data. Peers with low mobility and those located near the center of the region have a low probability of leaving the region in the near future, thus minimizing the average cost of each inter-region mobility.

5.1.4 Fault Tolerance and Replication

There is one major obstacle that all peer-to-peer systems must overcome: each peer can be expected to suddenly disconnect (or crash). Due to the high peer mobility and

low reliability of wireless links, the problem is more severe in the MP2P networks than Wired P2P networks. We must therefore make fault-tolerance a priority.

An efficient general solution to fault tolerance problem is impossible to construct, so we make three assumptions based on our target application and expected user behavior in MP2P networks. i) *Node failures are independent*: Since nodes are owned by different users and nodes use peer-to-peer wireless links, we assume that a set of nodes with adjacent geographic locations are highly likely to have independent failures. ii) *Most users will gracefully quit from the network*: If a node keeps some keys, the user will wait to transfer these keys to other nodes in the same region before disconnecting the node from the MP2P system. iii) *Message will be routed to the correct node*: Because of node mobility and inhomogeneous node distribution in MP2P networks, a route from source to destination node may not exist. If we consider a MP2P system with a high node density, it is reasonable to assume that messages eventually reach the correct node.

Most underlayer ad-hoc routing protocols provide limited fault-tolerance to make the routing resilient to user mobility and node failures, but data items in MP2P systems still need to be replicated to improve their availability. Furthermore, the replicas must be kept consistent with the original after each update. In PReCinCt, the keys of data items are kept at their home regions, a *home region failure* is said to occur when there is no copy of a key in its home region. Home region failure happens when there is no peer in a region, or a peer did not send back the key to the home region after an inter-region mobility, or sudden death occurs to the peer holding the key. Based on three assumptions, we design a lightweight data replication mechanism in PReCinCt to tolerate failures at the network, node and home region levels. The algorithm tries to keep at least one replica of all keys in a region other than the home region under all circumstances to improve the availability of each data item. Our discussion is based on a single replica, but can

be extended to multiple replicas to cope with higher failure frequencies, at the cost of increased bandwidth usage for replica messages and more cache space usage for replicas.

As we mentioned in the Data Search Process section, home region of each key is determined by selecting the region whose center location is closest to the key's hash value. Given a data item with key K whose hash value is the location L, then the home region R_h is the region whose center location (L_h) is closest to L. We will similarly select the next closest region R_r whose center location is L_r as a replica region of K, i.e., $\forall R_i \ : dist(L - L_h) \leq dist(L - L_r) \leq dist(L - L_i)$. This means that a request message for key K will always be routed to the region R_h. If there exists a network failure, a node failure or a region failure of R_h, this request message will instead be rerouted to the replica region R_r. After reaching the R_r, localized flooding is used to locate the requested data item. In order to maintain the consistency between original key and replica, update message need to be sent from home region to the replica region after the home region receives a data update for key K, which we will discuss in the next section.

5.2 Cooperative Caching in PReCinCt

The storage space of each peer is divided into two parts: static and dynamic caches. The static cache contains the values of keys that belong to the region where the peer currently resides. The dynamic cache, on the other hand, contains data items that are placed opportunistically at a peer to reduce latency of subsequent retrievals. The dynamic cache is optimally managed by a greedy cache mechanism. In this section we present the caching scheme employed in PReCinCt that includes a cumulative cache, a cache admission control and a cache replacement policy.

C: the set of data objects in cache;
$U(d_j)$: the unity function;
u_j: utility value of cached item j;
L: minimal utility in the cache;

Procedure Caching(d_j)
Begin
if d_j is retrieved from home region
  while there is not enough space for d_j;
    let $L = MIN_{k \in C}(u_k)$;
    Evict d_k such that $u_{jk} = L$;
  end while
  Add d_j into cache;
  $u_j = L + U(d_j)$;
else
  discard d_j;
End

Figure 5.2 The Caching Algorithm of PReCinCt Scheme.

5.2.1 Cumulative Cache

Caching data items in the local caches of peers, helps reduce latency and increase accessibility. When a peer P_i requests for data item d_j, it first attempts to obtain the data locally from its own region by broadcasting the request to its region members. If the request can be satisfied within the same region, a local hit occurs. If not, the request is sent to the home region of that data. If a peer along the path to the home region has the requested data item d_j, then it can serve the request without forwarding it further towards the home region. Otherwise, the request will be forwarded to the home region of item d_j. Since the local caches of the peers virtually form a cumulative cache, decisions regarding the caching of a data item and its eviction from the cache depends not only on the peer itself but also on the neighboring peers. We propose a cache admission control policy and a cache replacement algorithm for cumulative caches.

5.2.2 Cache Admission Control

When a peer P_i receives a response for the requested data item, a cache admission control is triggered at P_i to decide whether the data item should be cached. If the origin of the data resides in P_i's region, then the item is not cached; otherwise it is cached at the peer P_i. Peers cooperatively cache data and thus it is unnecessary to replicate data in the same region, as they can be obtained locally for subsequent requests.

5.2.3 Cache Replacement Policy

We have developed a greedy replacement policy that considers three factors while selecting a victim: i) the access count of the data items reflecting the popularity of a data item in the region; ii) the sizes of the data items; and iii) the region-distance which is the distance between the regions of the requesting and the responding peers. This algorithm incorporates the region-distance as an important parameter in selecting a victim for replacement. The greater the region-distance, the greater is the utility of the data. This is because caching data items which are further away, save bandwidth and reduce latency for subsequent requests. Thus we call our replacement algorithm *Greedy-Dual Least-Distance* (GD-LD).

The cache replacement policy uses a utility function to assign a utility value for each data item, based on the three factors mentioned above. Since the nodes in the region cooperatively cache data, the utility value reflects the importance of the data item to the entire region. The utility value is calculated using the following expression:

$$Utility = w_r \times ac_j + w_d \times reg\_dst + w_s \times \frac{1}{s_j} \tag{5.1}$$

where a_j is number of times the item d_j has been accessed in the region; reg_dst is the distance between the requesting region and the home region for the data; s_j is size of the item d_j, and w_r, w_d, w_s are the corresponding weight factors. The utility value of the

data item is updated when there is a hit. This value is used by the greedy replacement algorithm at the peer to find an optimal replacement and thereby manage the cache efficiently. Details of the caching algorithm are given in Figure 5.2.

5.3 Data Consistency in PreCinCt

In this section we discuss our proposed *Push with Adaptive Pull* scheme to maintain data consistency among replicas in the MP2P network. This scheme combines the push-based invalidation initiated by a peer updating a data item with the pull based invalidation initiated by the individual peers that maintain the cache. During the push phase the peer updating a data item sends the update to the home region for that data item. During the pull phase peers poll the home regions to check if their cache contents are obsolete. We propose an adaptive polling scheme which determines the frequency at which peers should poll the home regions.

Push Phase: When a peer P_{update} initializes an update request for a data item d_j at time t, P_{update} updates d_j and pushes the update message to the home region R_h of d_j as well as replica region R_r of d_j. Within the home and replica regions, flooding is used to locate the peer, say P_i, which has the data item d_j. When P_i is located, it updates its copy of data item d_j. Thus by pushing the update to only the home and replica regions instead of to the entire network, our scheme consumes less wireless bandwidth and incurs less message overhead to maintain data consistency. After receiving the update message, P_i updates the Time-to-Refresh (TTR) value for d_j. The TTR values are dynamically maintained to reflect the update rates for the data items. The algorithm of the Push phase is given in Figure 5.3.

Pull Phase: The pull phase of the algorithm is initiated by the individual peers interested in checking the validity of their cached items. During this phase peers poll

Notations:
d_j: updated data item;
t: time at which the update is initiated;
P_{ud}: the peer initiating an update for d_j;
R_j^h: home region for data d_j;
R_j^r: replica region for data d_j;
$M(d_j, t)$: Update message for data d_j which is updated at time t;

Procedure Push
Begin
When P_{ud} initiates an update message $M(d_j, t)$
Obtain the home region R_j^h and replica region R_j^r for d_j
Route the message $M(d_j, t)$ to regions R_j^h and R_j^r
For each P_i in R_j^h and R_j^r
If P_i have cached d_j
Update d_j and store the update time t in P_i
P_i calculates TTR of data item d_j
Else
Discard $M(d_j, t)$;
End

Figure 5.3 Push Phase.

the home regions to determine the freshness of data items in their cache. We propose an adaptive pull mechanism.

Adaptive Pull: The notion of adaptive polling has been used in the context of web cache consistency [31] and we adopt a similar idea here. In our proposed adaptive polling mechanism, peers vary the frequency with which they poll the home region for a data item, depending on the update rate for that data item. In this scheme the polling frequency is dynamically varied so that data items that are frequently updated are polled more often than data items that are relatively static.

The home regions assign a TTR value to each of their data items. This TTR value is varied dynamically by the home region to reflect the update rate for the associated data item. When the home region for data item d_j receives an update for d_j, it locates

the peer P_i which holds d_j. t_{upd_intvl} is the interval between successive updates of d_j. α $(0 \leq \alpha \leq 1)$ is a constant factor to weigh the relative importance of the recent and past updates, then the TTR value is updated as:

$$TTR = \alpha \times TTR + (1 - \alpha) \times t_{upd\_intvl} \tag{5.2}$$

This TTR value will be sent by the home region to other peers on subsequent requests for data item d_j. When a peer requests for d_j and there is a local cached copy, then it determines if the TTR of d_j has expired. The peer polls the home region only if the TTR of d_j has expired. Thus we avoid too many polls to the home regions while still ensuring good consistency. The detailed algorithm for pull phase is given in Figure 5.4.

Notations:
d_j: requested data item;
P_j^q: the peer requesting d_j;
R_j^h: home region for data item d_j;

Procedure Pull
Begin
When P_j^q requests data item d_j
If d_j is cached in P_j^q
  If TTR for d_j has expired
    Obtain the home region R_j^h for d_j
    Poll the home region R_j^h
    Inquire for missed updates based on the last update times
    Update the TTR and local cached copy
  Else
    Use the cached copy of d_j to satisfy the request
End

Figure 5.4 Pull Phase.

The main advantages of our *Push with Adaptive Pull* mechanism are: 1) The push messages need to be sent only to the home and replica regions, and not to the entire

network, thus saving bandwidth and power consumption; 2) As the frequency of polling is determined by the update rate for the data item, redundant polls to the home region are avoided. This saves the bandwidth as well as reduces latency.

5.4 Performance Analysis of the PReCinCt Scheme

In this section we briefly describe the energy model used in our performance evaluation and determine the energy consumption of the proposed PReCinCt.

Energy as a distributed network resource has unique properties that distinguish it from other resources [25]. Energy is non-renewable: a mobile node has a finite, monotonically decreasing battery energy. Unlike bandwidth the energy cost requires separate calculations with respect to the sender, the intended receiver and other nodes which overhear the message. This makes it necessary to distinguish between broadcast traffic which is processed by all nearby nodes and point-to-point traffic which is processed by the intended receiver(s) and discarded by all other nearby nodes. The details of the energy model for the analysis was presented in the last chapter in Section 4.3.1.

5.4.1 Analytical Model of the PReCinCt Search

In the light of the discussions above, we now present our analytical model for energy consumption of one query search of PReCinCt scheme. Before we present the analytical model, we give the assumptions used in the analysis and list the notations.

1. All nodes in the network are two-dimensionally Poisson distributed with density ρ, i.e., the probability $p(i, A)$ of finding i nodes in an area of size A is given by,

$$p(i, A) = \frac{(\rho A)^i e^{-\rho A}}{i!} \tag{5.3}$$

2. All nodes have the same transmission and receiving range, denoted by r. N is the average number of neighbor nodes within a circular region of radius r. Therefore, we have $N = \rho \pi r^2$.

3. The energy cost of data processing is negligible compared with the energy cost of data communication.

The notations used in the analysis are listed as follows.

- N: number of nodes in the network;
- M: number of regions;
- $l \times l$: size of the whole network area;
- r: transmission range of each MU;
- R_q: the set of MUs in requesting region;
- R_h: the set of MUs in home region;
- PH: the set of MUs in route path from requesting region to home region;
- P_i: mobile peer i;
- P_{ij}: the probability data item d_j is cached in P_i;
- s_q: size of one query message;
- $E_{bd\_sd}(s_q)$: energy cost to broadcast a query message;
- $E_{bd\_rv}(s_q)$: energy cost to receive a broadcast query message;
- $E_{p2p\_sd}(s_q))$: energy cost to unicast a query message;
- $E_{p2p\_rv}(s_q))$: energy cost to receive a unicast query message;

In the model, we divided the energy cost of a query search PReCinCt into three parts. 1) energy cost (E_1) due to localized broadcast in requesting region; 2) energy cost (E_2) due to route query to home region; and 3) energy cost (E_3) due to localized broadcast in home region;

According to the energy analytical model given in [78], during a query broadcasting in one region, each MU in the region will send the query once, and each MU receives multiple duplicated queries from all neighbors. Therefore, E_1 is calculated as follows.

$$E_1 = (1 - \prod_{MU_i \in R_q} (1 - P_{ij})) \left[\frac{N}{M} \times E_{bd\_sd}(sz_q) + \frac{\pi r^2 N}{l^2} \times \frac{N}{M} \times E_{bd\_rv}(sz_q) \right] \quad (5.4)$$

In the PReCinCt scheme, we use geographical hash table to deploy all data items in the whole network area, so that the probability of a region to be a home region of a data query is same for every region. Thus, the average length of route path from requesting region to home region is equal to the distance between two randomly picked regions in the network area, which is given by $\frac{\sqrt{2}}{2}l$. Therefore, E_2 is calculated as follows,

$$E_2 = \prod_{MU_i \in R_q} (1 - P_{ij})(1 - \prod_{MU_i \in PH} (1 - P_{ij})) \times \frac{\sqrt{2}l}{4r} \times (E_{p2p\_sd}(sz_q) + E_{p2p\_rv}(sz_q)) \quad (5.5)$$

If the query cannot be satisfied by the caches of requesting region and cumulative caches in route path, the query will reach the home region, where a localized broadcast is used to find the requested item. The calculation of E_3 is similar that of E_1, E_3 is calculated as follows.

$$E_3 = \prod_{MU_i \in R_q} (1 - P_{ij}) \prod_{MU_i \in PH} (1 - P_{ij}) \left[\frac{N}{M} \times E_{bd\_sd}(sz_q) + \frac{\pi r^2 N}{l^2} \times \frac{N}{M} \times E_{bd\_rv}(sz_q) \right] \quad (5.6)$$

Since the average cost of a query search consists of these three parts, the average cost of a PreCinCt search is given by,

$$\begin{aligned} E &= E_1 + E_2 + E_3 \\ &= (1 - \prod_{MU_i \in R_q} (1 - P_{ij})) \left[\frac{N}{M} \times E_{bd\_sd}(sz_q) + \frac{\pi r^2 N}{l^2} \times \frac{N}{M} \times E_{bd\_rv}(sz_q) \right] \\ &+ \prod_{MU_i \in R_q} (1 - P_{ij})(1 - \prod_{MU_i \in PH} (1 - P_{ij})) \times \frac{\sqrt{2}l}{4r} \times (E_{p2p\_sd}(sz_q) + E_{p2p\_rv}(sz_q)) \\ &+ \prod_{MU_i \in R_q} (1 - P_{ij}) \prod_{MU_i \in PH} (1 - P_{ij}) \left[\frac{N}{M} \times E_{bd\_sd}(sz_q) + \frac{\pi r^2 N}{l^2} \times \frac{N}{M} \times E_{bd\_rv}(sz_q) \right] \end{aligned}$$

5.5 Simulation Results

To measure the performance of our data retrieval and caching schemes, we simulated the algorithms on a variety of static and mobile network topologies. We focus mainly on the mobile simulation results in this chapter, as it is more demanding and challenging in MP2P networks. GPSR [33] is used as the wireless routing protocol. We have modified it to route to regions instead of specific destinations by forwarding the packet towards the center of the region and using broadcast inside the region. The performance metrics measured were average latency and energy consumption. We compare our cache replacement algorithm GD-LD with the well known GD-Size [14]. Likewise, we evaluate our cache consistency scheme namely *Push with Adaptive Pull* with the *Plain-Push* and *Pull-Every-time* schemes.

5.5.1 Simulation Environment

We simulated our experiments in NS-2 [108]. The NS-2 simulation model simulates nodes moving in an unobstructed plane. Motion follows the random waypoint model [11]. In our simulations the nodes are initially placed in a rectangular region of $1200m \times 1200m$ divided into equal sized regions. We conducted simulations for a network of up to 160 nodes with a nominal 250m transmission range and a wireless bandwidth of 11Mbps. The time interval between two consecutive requests and updates generated from a peer follows a Poisson distribution with a mean of 30 seconds. Each peer generates accesses to data items following a Zipf distribution with a skewness parameter, Θ. Nodes pause for a period of 5s between motion steps. We simulate maximum velocities of 2, 8, 12, 16 and 20m/s. The default number of regions is 9.

5.5.2 Simulation Experiments

The PReCinCt scheme is compared with the flooding and the expanding ring search schemes for energy consumption under varying node densities and moving speeds [47].

Three sets of experiments are reported in this chapter for evaluating the cooperative caching and consistency mechanisms of PReCinCt. In the first set of experiments the GD-LD cache replacement algorithm was compared with the GD-Size algorithm. The performance metrics measured were byte hit ratio and latency. The second set of experiments compare different cache consistency algorithms and show that the proposed *Push with Adaptive Pull* scheme performs better than *Plain-Push* and *Pull-Every-time* schemes. Finally the third set of experiments validates the theoretical studies by comparing the theoretical and simulation results.

5.5.2.1 Cache Replacement Experiments

In these set of experiments the developed cache replacement algorithm GD-LD is compared with GD-Size [14]. The simulations were conducted on a topology with 80 nodes moving at a speed of 6m/s.

Figure 5.5 shows the variation of latency to fetch a data item under varying cache sizes. GD-LD by far outperforms the GD-Size algorithm for all cache sizes. This is due to two reasons: (1) The developed algorithm includes a new parameter - region distance while calculating the utility value for an item. Region distance is the distance between the regions of the requesting and responding peer. The algorithm favors items that are further away as caching these items would reduce latency and save bandwidth for subsequent requests for the same item, and (2) The GD-Size algorithm, penalizes a large sized data item without considering its popularity or the cost of fetching it again for a subsequent request.

Figure 5.6 shows the byte hit ratio of the two caching algorithms. We observe GD-LD is able to achieve much higher byte hit ratios as compared to that of the GD-Size. This is because GD-Size favors small data items independent of their popularity-thus a large, popular data item stands less chance of being cached under GD-Size.

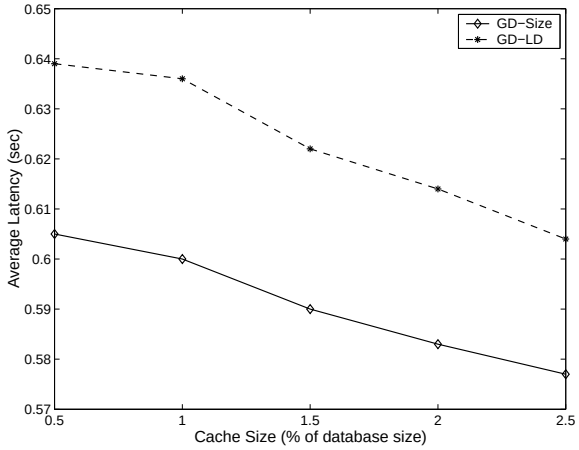

Figure 5.5 Variation of Latency with Cache Size.

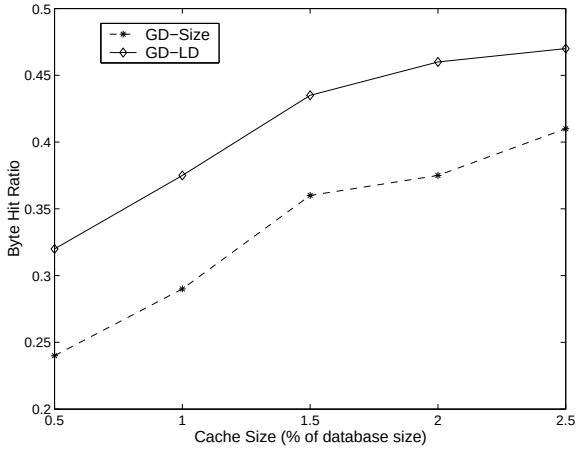

Figure 5.6 Variation of Byte Hit Ratio with Cache Size.

5.5.2.2 Data Consistency Experiments

In this section we compare the proposed *Push with Adaptive Pull* cache consistency algorithm with the *Plain-Push* and *Pull-Every-time* schemes. The performance metrics measured are: 1) Control Message Overhead, which is measured as the total number of messages generated in the network to maintain data consistency among the replicas; 2) False Hit ratio(FHR), measured as the ratio of the number of stale hits to the number of hits that are shown as valid; and 3) The average latency to retrieve a data item.

In the proposed model, the time interval between two consecutive requests, $T_{request}$ and the time between two consecutive updates, T_{update} generated from a peer follows a Poisson distribution. We measure the effects of the time between successive updates T_{update} on the different cache consistency schemes. We fix $T_{request}$ to 30 seconds, and vary T_{update} so that the ratio of $T_{update}/T_{request}$ varies from 1 to 5. A ratio of 1 indicates the highest update rate.

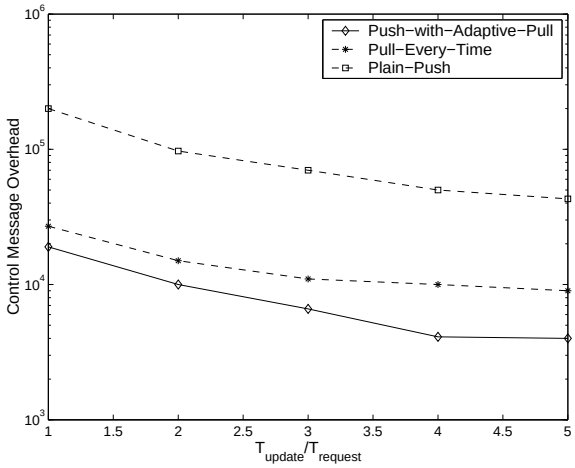

Figure 5.7 Effect of Update Rate on the Control Message Overhead.

Figure 5.7 shows the control message overhead of the three consistency schemes. The message overhead of all the schemes decreases with the update rate due to lesser invalidation messages that are generated. The overhead of *Plain-Push* is extremely high as the invalidation messages are flooded to the entire network. In the *Push with Adaptive Pull* scheme the invalidation messages are pushed only to the corresponding home region thus incurring almost 89% less message overhead than the *Plain-Push* scheme. The *Pull-Every-time* scheme also incurs higher overhead than that of the proposed scheme, as in the *Pull-Every-time* scheme the peers are required to poll the home regions for every data request to check for the validity of their cached item. In contrast the proposed scheme

polls the home region only when the TTR of that data item has expired, thus achieving 24%-57% less overhead than that of the *Pull- Every-time*. As the update rate increases the adaptive scheme tends to produce smaller TTR values, which results in slightly more polls.

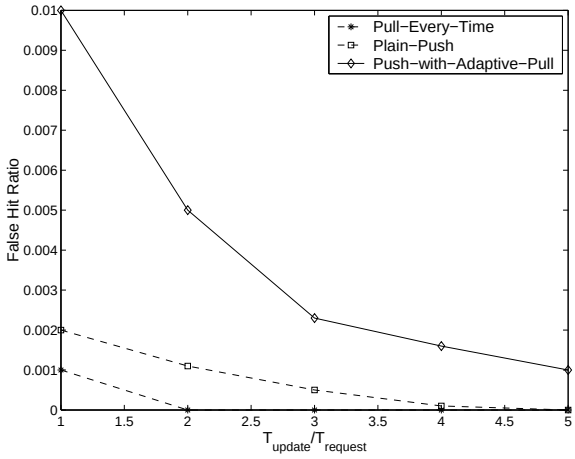

Figure 5.8 Effect of Update Rate on the False Hit Ratio.

The FHR of the three schemes is shown in Figure 5.8. We observe that the proposed scheme incurs the highest FHR, as the peers poll the home regions for data only when the TTR has expired, thus increasing the probability of a false hit. However, we see that this ratio is very small, 0.01 even with highest update rate. The*Plain-Push* scheme also incurs some false hits, as it is possible that the invalidation messages do not reach all the peers due to network congestion, network partition or peer mobility.

Figure 5.9 shows the average latency to retrieve a data item with varying update rates. It is observed that the *Pull-Every-time* scheme has the highest average latency, as in this scheme the peers are required to poll the home regions for every request which incurs an extra round-trip delay in obtaining the requested data.

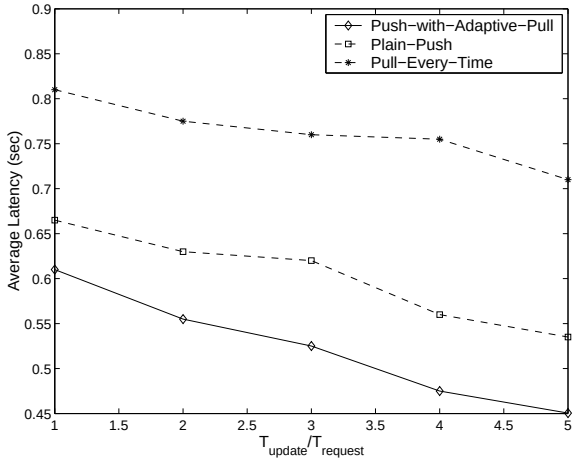

Figure 5.9 Effect of Update Rate on the Latency per Request.

5.6 Summary

Mobile Peer-to-Peer networks (MP2P) promise to offer ubiquitous information services to mobile users in the environments without wireless infrastructure. With more and more mobile users having location detection capability, we propose a novel location-aided data retrieval scheme, called PReCinCt, of our proposed framework for the peer-to-peer data searching in large scale mobile peer-to-peer system. The data retrieval scheme is scalable and incurs less overhead with peer mobility. The cooperative caching scheme enables peers in a region to share their data, thus providing a unified view of the cache. This helps in alleviating the message latency and limited accessibility problems in MP2P networks. In order to handle dynamic data, PreCinCt also includes an effective cache consistency scheme called *Push with Adaptive Pull* that incurs less control message overheads as compared to that of existing schemes.

In summary, the proposed framework integrated three different information access approaches, i.e., pull based, push based, peer-to-peer based to support ubiquitous information services under different wireless networks. In the framework, different *utility models* are developed to optimize cache management and cooperation for user data

accesses. The utility models considers various resource constraints (e.g., energy, bandwidth, memory, network connectivity etc) of mobile and distributed environments. The framework provides a flexible solution for information services in different application scenarios.

CHAPTER 6
CONCLUSION AND FUTURE WORK

Ubiquitous computing promises to provide the information access service whenever and wherever. While the wireless communication infrastructure is and will continue to be characterized by a heterogeneous multitude of systems. Next generation (4G) mobile systems focus on seamlessly integrating the existing wireless technologies. 4G networks are all-IP based heterogeneous networks that allow users to use any system anytime, anywhere. Users carrying an integrated terminal can use a wide range of applications provided by multiple wireless networks. Users can use multiple services from multiple service provides at the same time.

In order to support ubiquitous information services in the next generation mobile systems, in this dissertation, we develop a utility based resource aware framework to support ubiquitous information access in the heterogenous networks through data caching and peer-to-peer cooperative sharing. The framework considers the constrained resource of mobile devices and wireless networks and provides flexible, efficient and scalable data information services to mobile users. In this framework, *utility functions* are used to optimize cache management and get optimal performances for user data accesses under different wireless networks. The utility functions consider various resource constraints (e.g., energy, bandwidth, memory, network connectivity etc) of mobile and distributed environments. The proposed framework efficiently integrates various information providing models, (push-based, pull-based and peer-to-peer based), it can support large scale information services in mobile and distributed environments.

Particularly, tn the proposed framework, a novel energy and bandwidth efficient data caching mechanism, called GreedyDual Least Utility (GD-LU), is proposed to en-

hance dynamic data availability to mobile users. The proposed utility-based caching mechanism considers several characteristics of mobile distributed systems, such as connection-disconnection, mobility handoff, data update and user request patterns to achieve significant energy savings in mobile devices. Our comprehensive simulation experiments demonstrate that the proposed caching mechanism achieves more than 10% energy saving.

Moreover, a novel passive prefetching scheme is proposed to support information access in data broadcast networks. A threshold based on relative utility value of each data item is deployed to admit data items from broadcast channel. By choosing proper threshold value, passive prefetching can achieve near-optimal performance tradeoff between access latency and energy consumption.

In order to support information access in multi-hop hybrid wireless networks, we introduce a novel peer-to-peer information sharing scheme called *energy efficient peer-to-peer caching with optimal radius* (EPCOR). In EPCOR, a localized P2P overlay network is built among the mobile users to facilitate cooperative sharing of data based on network proximity and data preference. In order to conserve energy, each MU in localized overlay shares a data item in a *cooperation zone*. Both analytical and simulation results show that EPCOR enhances the peer-to-peer data sharing among mobile users and achieves significant performance improvement in power saving, network throughput and load distribution among mobile users in multi-hop wireless networks.

Finally, we investigate location-aided data retrieval in the mobile ad hoc networks and proposed a novel scheme, called *Proximity Regions for Caching in Cooperative MP2P Networks* (PReCinCt) to efficiently support scalable data retrieval in large-scale mobile ad hoc networks. In the PReCinCt scheme, the network topology is divided into geographical regions where each region is responsible for a set of keys representing the data. Each key is then mapped to a region location based on a geographical hash function. A geographic-

aided routing protocol is used to route traffic of each data retrieval. Moreover, a data replication scheme is used to handle fault tolerance problem in PReCinCt. In order to further save bandwidth for each data retrieval, PReCinCt incorporates a cooperative caching scheme that caches relevant data among a set of peers in a region. Simulation results show that PReCinCt can significantly save search cost in large scale mobile ad hoc networks compared to existing flooding based schemes.

All novel algorithms and functional modules of propose framework have been developed and thoroughly evaluated with comprehensive simulation experiments. To the best of our knowledge, the proposed work in this dissertation is the first comprehensive attempt to build a resource aware middleware framework for mobile devices to support ubiquitous information access in next generation heterogenous wireless networks.

The research work of this dissertation can lead to other future work as follows:

- Security management in peer-to-peer data sharing.

  Reputation based mechanism could be used in the peer-to-peer data sharing. A peer would receive a reputation based on the history of successful providing data to other peers. This trust management mechanism could to be used to motivate peers to share their information and could also be used to detect and exclude the malicious peers from the data sharing.

- Supporting real time multimedia streaming.

  Supporting multimedia applications is the major goal of next generation mobile systems. Although, we mainly focus on the discrete data services in this dissertation, caching and peer-to-peer sharing ideas of our work can be further studied to support real time multimedia applications in wireless networks.

- Context-aware information services.

  Context-aware computing is a ubiquitous computing paradigm in which applications can discover and take advantage of contextual information (such as user loca-

tion, time of day, and user activity). Although, we did not investigate context-aware information services in this dissertation. The proposed framework can be incorporated with research results of context-ware computing to provide context-aware information services to mobile users in next generation mobile systems.

- Prototype development.

  The middleware based framework needs proper programming model with appropriate application programming interfaces (API) for deployment. Because the main objective of this dissertation is to identify the functional modules and some systems parameter settings, we have left the design of the prototype system under scope of future works.

REFERENCES

[1] M. Abrams, C.R. Standbridge, G. Abdulla, S. Williams and E.A. Fox, "Caching Proxies: Limitations and Potentials," *WWW-4 Boston Conference*, December 1995.

[2] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik "Broadcast disks: Data management for asymmetric communications environments," *In Proceedings of ACM SIGMOD Conference on Management of Data*, pp. 199-210, San Jose, CA, May 1995.

[3] S. Acharya, S. Muthukrishnan,"Scheduling On-demand Broadcasts: New Metrics and Algorithms", *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, pp 43-54, October 1998.

[4] S. Acharya, M. Franklin, and S. Zdonik "Balancing push and pull for data broadcast", *In Proceedings of ACM SIGMOD Conference on Management of Data*, pp. 183-194, Tucson, AZ, May 1997.

[5] J. Al-Muhtadi, D. Michunas and R. Campbell, "A Lightweight Reconfigurable Security Mechanism for 3G/4G Mobile Devices," *IEEE Wireless Communication,*, 9(2), pp. 60-65, April 2002.

[6] D. Barbara and T. Imielinksi, "Sleepers and workaholics:Caching strategies for mobile environments," *Proceedings of ACM SIGMOD Conference on Management of Data*, Minneapolis, MN, pp.1-12, May 1994.

[7] P. Barford and M. Crovella,"Generating Representative Web Workloads for Network and Server Performance Evaluation," *Proceedings of the ACM SIGMETRICS Conference*, pp. 151-160, 1998.

[8] M. Bender, S. Chakrabarti, and S.Muthukrishnan"Flow and stretch metrics for scheduling continuous job streams", *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp 270-279, San Francisco, CA, January 1998.

[9] T. Berners-Lee, R. Fielding and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," *RFC-2396, MIT LCS and Xerox Co.,*, August 1998.

[10] L. Breslau, P. Cao, J. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-Like Distributions: Evidence and Implications," *In IEEE Proceedings of INFOCOM*, pp 126-134, 1999.

[11] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," *ACM MobiCom*, pp. 85-97, October 1998.

[12] E. Buracchini, "The Software Radio Concept," *IEEE Communication Magazine*, 38(9) pp. 138-143, 2000.

[13] P. Cao and C. Liu, " Maintaining Strong Cache consistency in the World Wide Web," *IEEE Transactions on Computers*, volume. 47, no. 4, pp. 445-457, Apr. 1998.

[14] P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms," *Proc. USENIX Symp. Internet Technologies and Systems*, pp. 193-206, Dec. 1997.

[15] G. Cao, "Proactive Power-Aware Cache Management for Mobile Computing Systems," *IEEE Transactions on Computers*, 51(6), pp. 608-621, June 2002.

[16] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *ACM Intl. Conf. on Computing and Networking (Mobicom)*, pp 200-209, August 2001.

[17] H. Chaskar and R. Koodli, "A Framework for QoS Support in Mobile IPv6," *draft-chaskar-mobileip-qos-01.txt(work in progress)*, March 2001.

[18] H. Che, Y. Tung, and Z. Wang,"Hierarchical Web Caching Systems: Modeling, Design, and Experimental Results," *IEEE Journal on Selected Areas in Communication*, 20(7), pp 1305-1315, 2002.

[19] A. Datta, M. Hauswirth, and K. Aberer, "Updates in Highly Unreliable, Replicated Peer-to-Peer Systems, " *In Proceedings of ICDCS 2003, 23rd International Conference on Distributed Computing Systems*, , pp. 76C85, Providence, Rhode Island, May 2003.

[20] H. Eguchi, M. Nakajima, and G. Wu, "Signaling Schemes over a Dedicated Wireless Signaling System in the Heterogenous Network," *Proc. IEEE VTC .,*, pp. 464-467, Spring 2002.

[21] ETSI "Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to Mobile, Portable and Fixed Receivers", *ETS 300 401*, 2nd Edition, May 1997.

[22] ETSI "Digital Video Broadcasting (DVB); A Guideline for The Use of DVB Specifications", *TR 101 200* , Sep. 1997.

[23] L. Fan, P. Cao, W. Lin, and Q. Jacobson, "Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance," *Proceedings of the ACM SIGMETRICS Conference*, pp. 178-187, June 1999.

[24] L. Fan, P. Cao, J Almeida, and A. Broder, "Summary Cache: A Scalable Wide Area Web Cache Sharing Protocol," *ACM SIGCOMM*, pp. 254-265, 1998.

[25] L. Feeney, " An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks, " *In ACM Journal on Mobile Networks and Applications*, 6(3), pp. 239-249, June 2001.

[26] L. Feeney, and M. Nilsson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad hoc Networking Environment," *In IEEE Proceedings of INFOCOM*, pp. 1548-1557, April 2001.

[27] FIPS 180-1, "Secure Hash Standard," *U.S. Department of Commerce/NIST National Technical Information Service,*, Springfield, VA, April 1995.

[28] P. Gauthier, D. Harada, and M. Stemm, " Reducing power consumption for the next generation of PDAs: It's in the network interface, " *In Proc. of MoMuC'96*, September 1996.

[29] S. Gitzenis and N. Bambos, "Power-Controlled Data Prefetching/Caching in Wireless Packet Networks," *In IEEE Proceedings of INFOCOM 2002*, pp. 1405- 1414, New York, June 2002.

[30] Allan Gut, "An Intermediate Course in Probability," Springer-Verlag New York, 1995.

[31] J. Gwertzman and M. Seltzer, " World-Wide Web Cache Consistency, " *In Proceedings of the 1996 USENIX Technical Conference*, pp. 141-151, January 1996.

[32] A. Kahol, S. Khurana, S.K.S. Gupta and P.K. Srimani,"A Strategy to Manage Cache Consistency in a Distributed Mobile Wireless Environment," *IEEE Trans. on Parallel and Distributed Systems*, 12(7), pp. 686-700, 2001.

[33] B. Karp, and H. T. Kung, " GPSR: Greedy Perimeter Stateless Routing for Wireless Networks, " *In Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000)*, pp.243-254, August 2000.

[34] R. Karrer, A. Sabharwal, and E. Knightly, "Enabling Large Scale Wireless Broadband: The Case for TAPs" *ACM SIGCOMM Computer Communication Review*, 34(1), pp. 27-32, 2004.

[35] W. Kellerer, H. Vogel and K. Steinberg, "A Communication Gateway for Infrastructure-Independent 4G Wireless Access," *IEEE Commnucations Magazine*, pp. 126-131, March 2002.

[36] W. Kellerer, P. Sties, and J. Eberspacher,“ IP Based Enhanced Data Casting Services Over Radio Broadcast Networks”, *Proc. IEEE ECUMN,*, pp 195-203, Colmar, France, October 2000.

[37] S. Khanna and V. Liberatore“ On Broadcast Disk Paging”, *SIAM J. Computing*, 29(5),pp. 1688-1702, 2000.

[38] U. Kubach, and K. Rothermel, “Exploiting Location Information for Infostation-Based Hoarding” *In ACM Intl. Conf. on Computing and Networking (Mobicom)*, pp. 15-27, 2001.

[39] Z. J. Haas and M.R. Pearlman,“The performance of query control schemes for the zone routing protocol,” *ACM/IEEE Thransactions on Networking*, 9(4), pp. 427-438,2001.

[40] T. Hara, “Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility,” *In IEEE INFOCOM*, pp. 1568-1576, April 2001.

[41] H-Y. Hsieh and R. Sivakumar, “On Using Peer-to-Peer Communication in Cellular Wireless Data Networks,” *IEEE Tran. on Mobile Computing*, 3(1), pp. 57-72, Jan-Mar 2004.

[42] Q. Hu and D.K. Lee, “Cache algorithms based on adaptive invalidation reports for mobile environments”, *Cluster Computing*, pp 39-50, 1998.

[43] S.Y. Hui, and K.H. Yeung, ”Challenges in the Migration to 4G Mobile Systems”, *IEEE Communications Magazine*, pp. 54-59, December 2003.

[44] S. Iyer, A. Rowstron, and P. Druschel, “Squirrel: A Decentralized Peer-to-peer Web Cache” *Proc. 21st Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 213-222, 2002.

[45] Z. Jiang and L. Kleinrock, “An Adaptive Network Prefetch Scheme,” *IEEE Journal on Selected Areas in Comminications (JSAC)*, 16(3), pp. 358-369, April 1998.

[46] J. Jing, A. Elmagarmid, A. Heal, and R. Alonso. "Bit-sequences: an adaptive cache invalidation method in mobile client/server environments", *Mobile Networks and Applications*, pp 115-127, 1997.

[47] M.S. Joesph, M. Kumar, H. Shen and S.K. Das, "Energy Efficient Data Retrieval and Caching in Mobile Peer-to-Peer Networks", *Workshop on Mobile Peer-to- Peer Computing, Third International Conference on Pervasive Computig and Communications*, pp. 50-54, Kauai, Hawaii, March 2005.

[48] G. Kortuem, J. Schneider, D. Preuitt, T.G.C. Thompson, S. Fickas, and Z. Segall, " When peer-to-peer comes face-to-face: collaborative peer-to-peer computing in mobile ad-hoc networks," *In Proceedings of the First International Conference on Peer-to-Peer Computing*,pp. 75-82, August 2001.

[49] W. H. O. Lau, M. Kumar, and S. Venkatesh, "A Cooperative Cache Architecture in Supporting Caching Multimedia Objects in MANETs," *The Fifth International Workshop on Wireless Mobile Multimedia*, pp. 56-63, 2002.

[50] T.H Le and A.H Aghvami, "Performance of an Accessing and Allocation Scheme for the Download Channel in Software Radio," *Proc. IEEE Wireless Commun. and Net. Conf.,*, vol. 2, pp. 517-521, 2000.

[51] M. Lindsey, M. Papadopouli, F. Chinchila and A. Sing,"Measurement and analysis of the spatial locality of wireless information and mobility patterns in a campus," *Techical Report TR03-006, Department of Computer Science, The University of North Carolina at Chapel Hill*, March 2003.

[52] C.W. Lin and D. L. Lee "Adaptive data delivery in wireless communication environments", *In Proceedings of the 20th IEEE International Conference on Distributed Computing Systems (ICDCS)* , pp. 444-452, Taipei, Taiwan, April 2000.

[53] S. Lim, W. C. Lee, G. Cao, and C. R. Das, "A Novel Caching Scheme for Internet based Mobile Ad Hoc Networks," *In IEEE Int'l Conf. on Computer Comminications and Networks (ICCCN)*, pp.38-43, October 2003.

[54] B. Li, M.J. Golin, G. F. Iaiano, and X. Deng, "On the Optimal Placement of Web Proxies in the Internet," *In IEEE INFOCOM*, pp. 1282-1290, New York, March 1999.

[55] B. Liu, Z. Liu, and D. Towsley, "On the Capacity of Hyrbid Wireless Networks," *In IEEE INFOCOM*, pp. 1543- 1552, March 2003.

[56] Y. D. Lin and Y.C. Hsu, "Multihop Cellular: A New Architecture for Wireless Communications" *In IEEE Proceedings of INFOCOM*, pp. 1273-1282, March 2000.

[57] P. Linga, I. Gupta, and K. Birman, "A Churn-Resistant Peer-to-peer Web Caching System" *ACM Workshop on Survivable and Self-Regenerative Systems*, pp. 1 - 10, October 2003.

[58] P. Lorenzetti, L. Rizzo and L. Vicisano, "Replacement Policies for a Proxy Cache," *http://www.iet.unipi.it/luigi/research.html.*

[59] H. Luo, R. Ramjee, P. Sinha, L. Li and S. Lu "UCAN: A Unified Cellular and Ad-Hoc Network Architecture" *In ACM Intl. Conf. on Computing and Networking (Mobicom)*, pp. 353-367, 2003.

[60] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker "Search and replication in unstructured peer-to-peer networks," *In Proceedings of the 16th annual ACM International Conference on Supercomputing*, pp. 84 - 95, June 2002.

[61] R. N. Mayo and P. Ranganathan, "Energy Consumption in Mobile Devices: Why Future Systems Need Requirements-Aware Energy Scale-Down," *Hp Labs Technical Report HPL-2003-167*, 2003.

[62] S-Y. Ni, Y-C. Tseng, Y-S. Chen, and J-P Sheu "The Broadcast Storm Problem in a Mobile Ad Hoc Network," *In Proceedings of the 5th annual ACM International Conference on Mobile Computing and Networking (MobiCom)*, pp. 151 - 162, Seattle, WA, August 1999.

[63] P. Nuggehalli, V. Srinivasan, and C. F. Chiasserini "Energy-Efficient Caching Strategies in Ad Hoc Wireless Networks" *In Proceedings of ACM MobiHoc*, pp. 25-34, 2003.

[64] M. Papadopouli, and H. Schulzrinne "Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices" *In Proceedings of ACM MobiHoc*, pp. 117-127, 2001.

[65] C. Perkins, and P. Bhagwat,"Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for mobile computers," *ACM SIGCOMM*, pp. 234-244, 1994.

[66] C. Perkins, and E.M. Royer,"Ad-hoc On-demand Distance Vector routing," *Proc. of WMCSA 1999*, pp. 90-100, 1999.

[67] C. Perkins, "IP Mobility Support for IPv4," IETF RFC 3344, August 2002.

[68] G.J. Pottie and W.J. Kaiser, "Wireless Integrated Network Sensor," *In Communications of the ACM*, 43(5), pp. 551-558, May 2000.

[69] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, " A scalable content addressable network," *In Proc. of ACM SIGCOMM*, pp. 161-172, San Diego, CA, August 2001.

[70] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initation Protocol," IETF RFC 3261, June 2002.

[71] A. Rowstron, and P. Druschel,"Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems", *Proc. of IFIP/ACM Intl. Conf. on Distributed Systems Platforms (Middleware 2001)*, pp. 329 - 350, Heidelberg, Germany November 2001.

[72] A. Rousskov and H. Schulzrinne, "Cache Digests," *Computer Networks and ISDN Systems*, pp. 22-23, 1998.

[73] L. Rizzo and Vicisano," Replacement policies for a proxy cache," *IEEE/ACM Trans. on Networking* , Vol.8, pp 158-170, April 2000.

[74] R. Rivest, "The MD5 Message-Digest Algorithm," *RFC-1321, MIT LCS and RSA Data Security, Inc,*, VA, April 1992.

[75] F. Sailhan and V. Issarny, "Cooperative Caching in Ad Hoc Networks" *In Intl. Conf. on Mobile Data Management (MDM)*, pp. 13-28, 2003.

[76] P. Sarkar, and J. H. Hartman, " Hint-based cooperative caching," *ACM Transactions on Computer Systems*, volume 18, number 4, pp. 387-419, 2000.

[77] H. Shen, M. Kumar, S. K. Das, and Z. Wang, "Energy-Efficient Data Caching and Prefetching for Mobile Devices Based on Utility," *ACM Journal of Mobile Networks and Applications*, Vol. 10, No. 4, pp. 465 - 474, January 2005.

[78] H. Shen, S. K. Das, M. Kumar, abd Z. Wang, "Energy-Efficient Peer-to-Peer Caching with Optimal Radius in Multi-Hop Hybrid Networks," *IEEE Trans. on Mobile Computing*, under second round review.

[79] H. Shen, M. Kumar, S. K. Das, and Z. Wang, "Energy-Efficient Caching and Prefetching with Data Consistency in Mobile Distributed Systems," *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Santa Fe, NM, April 2004.

[80] H. Shen, M. S. Joseph, M. Kumar, and S. K. Das, "PReCinCt: A Scheme for Cooperative Caching in Mobile Peer-to-Peer Systems," *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Denver, CO, April 2005.

[81] H. Shen, S. K. Das, M. Kumar and Z. Wang, "Cooperative Caching with Optimal Radius in Hybrid Wireless Networks," *Proceedings of Third International IFIP-TC6 Networking Conference (Networking 2004)*, LNCS, Vol. 3042, pp. 841-853, Athens, Greece, May 2004.

[82] J. Shim, P. Scheuermann and R. Vingralek, "Proxy Cache Design: Algorithms, Implementation and Performance," *IEEE Trans. on Knowledge and Data Engineering(TKDE)*, 11(4): 549-562, July/August 1999.

[83] A. C. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," *Proceedings of International Conference on Mobile Computing and Networking*, pp 155- 166, 2000.

[84] K. Stathatos, N. Roussoppulos, and J. S. Baras "Adaptive data broadcast in hybrid networks", *In Proceedings of the 23th International Conference on Very Large Data Bases (VLDB'87)* , pp. 326-335, Athens, Greece, August 1997.

[85] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, " Chord: A Scalable Peer-ot-Peer Lookup Services for Internet Applications," *In Proc. of ACM SIGCOMM*, pp. pp. 149-160, San Diego, CA, August 2001.

[86] C. Su and L. Tassiulas," Joint broadcast scheduling and user's cache management for efficient information delivery", *ACM Journal on Wireless Networks*, pp. 279-288, 2000.

[87] N.J. Tuah, M. Kumar, S. Venkatesh, and S.K. Das, "Performance Optimization Problem in Speculative Prefetching", *IEEE Trans. on Parallel and Distributed Systems*, 13(5), pp. 471-484, May 2002.

[88] N.H. Vaidya and S. Hameed "Scheduling data broadcast in asymmetric communication environments," *ACM/Baltzer Journal of Wireless Network (WINET)*, 5(3), pp. 171-182, 1999.

[89] Z. Wang, S. K. Das, H. Che and M. Kumar, "SACCS: Scalable Asynchronous Cache Consistency Scheme", In *Proceedings of ICDCDS Internation Workshop on Mobile Wireless Networks*, pp. 797-802, May, 2003.

[90] Z. Wang, M. Kumar, S. K. Das and H. Shen, "Investigation of Cache Maintenance Strategies for Multi-Cell Environments," *International Conference on Mobile Data Management (MDM)*, Australia, Melbourne, pp. 29-44, January 2003.

[91] Z. Wang, S. K. Das, H. Che and M. Kumar, "A Scalable Asynchronous Cache Consistency Scheme (SACCS) for Mobile Environments", *IEEE Trans. Parallel Distributed Systems*, 15(11), pp. 983-995, 2004.

[92] X. Y. Wang, W. S. Ng, B. C. Ooi, K. L. Tan, and A. Y. Zhou, "BuddyWeb: a P2P-based Collaborative Web Caching System" *Proc. International Workshop on Peer-to-Peer Computing*, pp. 247-251, 2002.

[93] M. Weiser, " Some Computer Science Issues in Ubiquitous Computing," *Communication of the ACM*, 36(7), pp. 74-84, 1993.

[94] D. Wssels, and K. Claffy, "ICP and the Squid Web Cache," *IEEE Journal on Selected Areas in Communication*, pp 345-375, 1998.

[95] S. Williams, M. Abrams, C.R. Standbridge, G. Abdulla and E.A. Fox, " Removal Policies in Network Caches for World-Wide Web Documnets," *Proceedings of ACM SIGComm96*, pp. pages 293-305, August 1996.

[96] R. Wooster and M. Abrams, "Proxy caching that estimates edge load delays," *Proc. 6th Int. World Wide Web Conf.*, pp. 977-986, Santa Clara, CA, April 1997.

[97] J.W. Wong "Broadcast Delivery," *Proceedings of the IEEE*, 76(12), pp. 1566-1577, December 1988.

[98] H. Wu, C. Qiao, S. De, and O. Tonguz, "Intergrated Cellular and Ad Hoc Relay systems: iCAR" *IEEE Journal on Selected Areas in Communications (JSAC)*, 19(10), pp. 2105-2115, Oct. 2001.

[99] K.L. Wu, P.S. Yu and M.S. Chen, "Energy-efficient caching for wireless mobile computing", *In 20th International Conference on Data Engineering*, pp. 336-345, 1996

[100] J. Xu, Q. Hu, W. Lee, and D. L. Lee, "Performance Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination," *IEEE Tran. on Knowledge and Data Engineering*, 16(1), pp. 125-139, January 2004.

[101] L. Yin, G. Cao, C. Das and A. Ashraf, "Power-Aware Prefetch in Mobile Environments," *IEEE International Conference On Distributed Computing Systems (ICDCS)*, pp. 571-578, July 2002.

[102] L. Yin, and G. Cao, "On Supporting Cooperative Cache in Ad Hoc Networks," *In IEEE Proceedings of INFOCOM*, pp. 2537-2547, March 2004.

[103] N.E. Young, "The K-Server Dual and Loose Competitiveness for Paging," *Algorithmica*, 11(6), pp. 525-541, 1994.

[104] IEEE 802.11 Working Group, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, 1999.

[105] The Gnutella Homepage. http://gnutella.wego.com/.

[106] http://www.pdos.lcs.mit.edu/roofnet.

[107] The Internet Engineering Task Force, "http://www.ietf.org".

[108] ns Notes and Documentation, "http://www.isi.edu/nsnam/ns/".

BIOGRAPHICAL STATEMENT

Huaping Shen was born in Dongyang, Zhejiang Province, China. He received his Doctorate in the Department of Computer Science and Engineering from the University of Texas at Arlington (UTA). Prior to that he received his B.S. and M.S. degrees from Fudan University, China, in 1998, and 2001, respectively, all in Computer Science. His current research interest is in the area of mobile wireless networks, peer-to-peer networks, distributed systems and pervasive computing. He has published more than eight papers in journals and international conferences in these areas. He received Texas Telecommunications and Engineering Consortium (TxTEC) scholarship, NOKIA scholarship and Rudolf Hermanns Graduate Fellowship from the University of Texas at Arlington.