Mobile Map: A Case Study in the Design & Implementation of a Mobile Application

Jane (Juanli) Liu

A thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in Information and Systems Science

Department of Systems and Computer Engineering

Carleton University

1125 Colonel By Drive

Ottawa, Ontario, K1S 5B6

Canada

The undersigned recommend to the

Faculty of Graduate Studies and Research

Acceptance of the thesis:

Mobile Map: A Case Study in the Design & Implementation of a Mobile Application

submitted by

Jane (Juanli) Liu

in partial fulfillment of the requirements

for the degree of

M.Sc. in Information and Systems Science

Professor T. Kunz, Thesis Supervisor

Chair, Department of Systems and Computer Engineering

Carleton University

ABSTRACT

Mobile computing is becoming more and more popular and common. In the near future, people will be able to use mobile computers (Laptops, palmtops, etc.) to do various computing tasks and to access the Internet from anywhere and anytime at will. The development of wireless facilities and mobile computers has made this goal possible, and some of the mobile applications exist already. But the situation is still far from ideal. There is still a lot of work to be done to reach the goal. Compared with a wired network environment, a mobile computing network has a lot of problems. A wireless link has low bandwidth and is unstable. Mobile devices are resource limited, like memory, storage, display area, power, etc. All these factors cause problems for mobile applications. Mobile network users usually suffer from long latencies and frequent disconnection, mobile applications suffers from bad performance. In recent years, a lot of research have been done to deal with these issues, to improve the mobile application performance. The proposed techniques include caching, prefetching, compression, etc.

The goal of this thesis is to study the techniques to improve client performance in client/server mobile computing systems, to identify which of the techniques are most promising and can be adapted to a tourist application (map viewing on small handheld devices, or other information like buses, menus, and theatre agendas, traffic and weather, etc.). We designed and implemented a mobile client/server application – a Mobile Map System. Based on our study on mobile computing techniques and the features of the mobile map application, we identified some caching and prefetching techniques and integrated them into our mobile map application. Through experiments, we studied the impact of these techniques to the performance of the application and concluded that the caching only method is a better solution to improve the performance of the mobile map application.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my thesis supervisor, Dr. Thomas Kunz, for his consistent support, invaluable guidance and suggestions during the course of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Mobile Computing Background

With the advent of cellular technology and portable computers we are on the verge of a new computing paradigm. This computing paradigm is now widely known as “Mobile” or “nomadic” computing [AHSON98].

In recent years Computer and Communication technologies have been developing fast. Using computers and accessing network information resources have become a necessary part of our daily work and daily life. There are numerous computers around the world that are connected by various kinds of networks. There are numerous applications that allow people to do almost everything they want. But, this situation is mostly restricted to users at fixed locations with static desktop computers and static wired networks.

There is still a lot of time we are mobile – moving among offices, homes, planes, trains, automobiles, conference rooms, classrooms and such. There is need to access computing resources not only when stationary but also while mobile or in transit [WEIS91].

As technology improves in the area of wireless facilities and mobile computers, mobile computing has become feasible. As of today, a variety of advanced mobile devices, some mobile wireless systems and mobile computing applications exist already. For example, people can send and receive emails and access Internet web sites using mobile computers via wireless networks. The future trend of telecommunication is to extend the telecom and computing services to mobile users, to break the restriction of user locations, to allow people access to computing resources anywhere and anytime at will. However, this is not a trivial task. Compared to static systems, mobile computing systems are constrained in important ways. These constraints are intrinsic to mobility, and are not just artifacts of current technology [SATY]. Mobile elements are resource-poor relative to static elements. Wireless links have low bandwidth and are unstable. Mobile elements must operate under a much broader range of networking conditions. The nature of wireless communication media and the mobility of computers combine to create fundamental new problems in networking, operating systems, and information systems [IMIE96].

Mobile computing is still at its early stage. In recent years there has been a lot of research in the mobile computing area. It has still a long way to go to reach the goal of connecting anywhere and anytime.

1.2 A Vision of the Future

The network of the future will be a unified one that is composed of different kinds of links, nodes, access points, and user devices. They could be wired links, wireless links, wired nodes, wireless nodes, wired access points, wireless access points. The user devices could be desktops, laptops, and palmtops. Different situations have different connection interfaces. A user can access the network in different ways depending on his location. He can access the network through the docking station in the office, through wireless LAN when roaming in a building, through cellular network when on the road. Abundant mobile computing applications will allow people to realize the dream of connecting to the network anytime and anywhere and to access computing resources at will.

1.3 Problems and Challenges

As said in Section 1.1, mobile computing is a promising area with still a lot of work to do. Compared with static systems, a mobile system has its intrinsic constraints that raise many fundamental problems. The many issues to be dealt with stem from three essential properties of mobile computing: Communication, mobility and portability [FORM94].

- **Wireless Communication**

Wireless connectivity is highly variable in performance and reliability. Compared with wired communication, wireless communication is characterized by lower bandwidth, higher error rates and more frequent disconnection. These factors cause long latency in transmitting data and high cost and frequent disconnection for users. Wireless technology has been developing to provide high speed wireless connection. Some buildings may offer reliable, high-bandwidth wireless connectivity while others may only offer low-bandwidth connectivity. Outdoors, a mobile client may have to rely on a low-bandwidth wireless network with gaps in coverage.

Security is another important issue in wireless communication because of the ease of connection to a wireless link. Hence appropriate security measures have to be taken. This is further complicated when users have access to cross security domains.

- **Mobility**

Mobile systems have to support mobility of mobile users. As mobile users move, they encounter networks with different features. Mobile users expect to be able to access the network at any time and from anywhere and to transparently remain connected and continue to use the network as they move about. A mobile computer must be able to switch from infrared mode to radio mode as it moves from indoors to outdoors, switch from cellular mode of operation to satellite mode as the computer moves from urban to rural areas. Mobile systems should support users moving across cells and different networks without interrupting services. This poses challenges in developing network protocols and management systems for mobile computing systems. Mobile systems and applications have to be designed to be adaptive to the change of the communication environment.

- **Portability**

To be portable, mobile computers must be small and light. The result is that mobile computers are typically resource-poor relative to static desktop computers. They have low processor speed, small memory space, small display screen, and limited battery power. Due to these constraints, a lot of desktop systems and applications cannot port to mobile computers directly. When designing mobile systems and applications for mobile computers, these factors must be taken into consideration.

The obvious and direct results of the above constraints are that mobile computing applications suffer from bad performance and mobile users suffer long latency and frequent disconnection. Research efforts are needed to address these issues.

1.4 Thesis Objectives

This thesis focuses on studying techniques to improve client performance in client/server mobile computing systems. The objectives of this thesis are:

1. Carry out a literature review on existing techniques and research on improving performance of mobile computing applications, to identify advantages and disadvantages of each technique.
2. Develop a new client/server mobile computing application- a Mobile Map System.
3. Based on the techniques studied and the features of the mobile map application, identify the most promising techniques and integrate them into the mobile map application. Through experiments, study the effects of these techniques to the application, to find out which technique or combination gives a better solution.

1.5 Thesis Outline

The rest of the thesis is structured as follows. Chapter 2 gives an introduction of the general concepts and theory of mobile computing. In Chapter 3, existing techniques and research for improving performance of mobile computing application are reviewed. The advantages and disadvantages of various techniques are discussed. Chapter 4 describes the design and development process of a new mobile computing application – a client/server mobile map system. In Chapter 5, some promising techniques are selected and integrated in the mobile map application. Through experiments, the effects of these techniques to the mobile map application are studied. Chapter 6 concludes this thesis and provides recommendations for future work.

CHAPTER 2

MOBILE COMPUTING

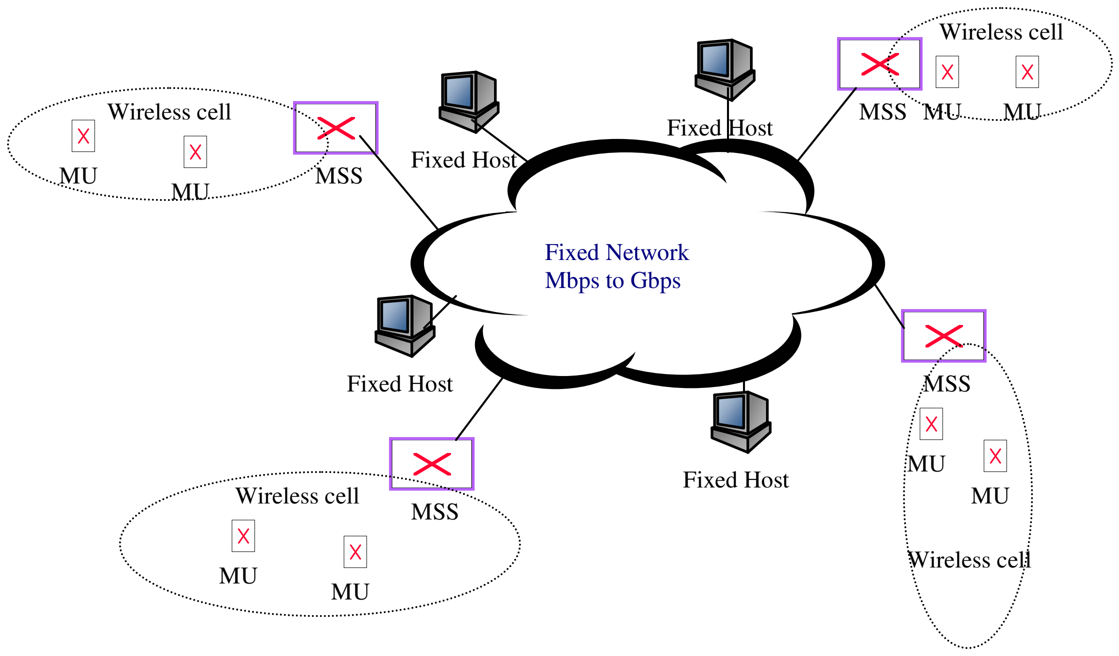
This chapter provides a background introduction to mobile computing. We introduce the main components, architecture, wireless technologies and services, and software systems of a mobile computing system, which form the foundation and living environment of mobile applications. The features of the underlying mobile network systems and mobile devices have fundamental impact on the performance of mobile computing applications, on the selection of techniques and the effectiveness of the chosen techniques to the applications.

2.1 Mobile Computing Systems – Architecture and Main Components

A typical mobile computing system consists of the following components [RACH96].

- A static communication backbone using wired means
- A set of static hosts
- Sets of mobile hosts (MHs) that can communicate with a predetermined static host
- Mobile subnets/wireless cells

The following figure displays the architecture of a general mobile computing system.



MU:	Mobile Unit
MSS:	Mobile Support Stations

Figure 2.1. Architecture of Mobile Communication Networks

The elements in a mobile computing system fall into two distinct sets: mobile units and fixed hosts. Some of the fixed hosts, called Mobile Support Stations, are augmented with a wireless interface to communicate with mobile units which are located within a radio coverage area called a cell. A cell could be a real cell as in cellular communication network or a wireless local area network that operates within the area of a building. In the former case the bandwidth will be severely limited (supporting data rates on the order of 10 to 20 Kbps). In the latter, the bandwidth is much wider – up to 10 Mbps. Fixed hosts will communicate over the fixed network, while mobile units will communicate with other hosts (mobile or fixed) via a wireless channel [BARB99].

2.2 Mobile Devices

The mobile device markets are developing very fast in recent years. The earliest mobile devices are simple function cell phones for voice and PDAs (Personal Digital Assistants) for personal data. As technology improves, more and more functions are added to the small user devices and new products are created. As of today, a variety of advanced mobile devices are already available offering touch screens, increased processing power, memory capacity, and battery lifetimes, and providing open interface based on mobile operating systems. New features such as GPS allow for location awareness and enable adaptive applications [FASB99]. These devices provide both computing and communication capabilities.

In today's market, mobile devices include notebook (laptop) computers, handheld computers, palmtop computers, cellular phones and hybrid new products. Laptop computers share a considerable common ground with their desktop counterparts in terms of computing resources such as memory and computing power. They can be carried on business trips, to conferences, to be used on airplanes and hotel rooms. They provide a lot of convenience to business people on trips. But they are still too big to carry around all the time. Handheld computers are smaller and lighter, but more limited in their functionality than laptop computers. Windows CE is the most popular operating system and although incompatible with standard Windows software, it does have miniature applications to help improve productivity (Pocket Word and Porket Excel) [ENRO01]. A palmtop computer is much more than a personal organizer. Palm-sized computers give you the ability to install additional software. The Palm OS (operating system) features a wide variety of third-party applications, as does Microsoft’s palm-sized OS, Windows CE. While Palm and Microsoft provide the OS, many manufacturers provide the hardware. Handspring, IBM, and the Palm itself all make hardware that runs the Palm OS while Casio, Hewlett-Packard, and Compaq have been among Windows CE’s biggest supporters. Palm OS is simple to use and inexpensive. Compared to Windows CE devices, Palm devices generally have slower processors and less RAM. Windows CE can perform multimedia tasks. The newest Windows CE devices come with software for playing digital music, reading electronic books, and organizing your finances.

The cell phones is basically for voice communication. Cell phones come in all shapes and sizes from various manufacturers. In general, they are smaller but more power consuming than PDAs because of cell phone’s RF power requirements. We summarize the features of different mobile device products in Table 2.1.

New hybrid products include smart phones that are turning into tiny computers and smart handhelds that are taking more and more functions of desktop computers. New handheld devices are small, light with a color screen. They have one or more expansion slots to create multiple expansion capabilities. You can insert a game card, wireless modem, GPS receiver, digital cameras, you can use Excel and Word, read eBooks, view video clips and photos, and browse the Web. The newest features in portable technology devices include Internet and email access, video and music transmission. Some of these features have been around for a while but the trend is that everything is getting smaller, smarter and faster [ENRO01].

Table 2-1. Mobile Devices

Device Type	Processing Power	Memory/ Storage	Screen Size	Weight	Communication Capability
Cell Phone	Few kHz	Few KB	Text/B&W	light	Voice-dial
Smart Phone	10-20 MHz	1-8 MB	320*240 Gray / Color	light	Voice-dial
Palmtop (Palm OS)	~20 MHz	8 MB	160*160 Gray / Color	<1 lb	Modem, serial port, dorking port, infrared port
Handheld (Win CE)	~100 MHz	32 MB	640*240 Gray / Color	1-2 lb	Modem, serial port, dorking port, infrared port
Laptop	>400 MHz	Hundreds of MB	1024*768 Color	6-7 lb	Modem, Ethernet Port, serial/parallel port, infrared

2.3 Wireless Technologies and Services

Wireless networks are the bridges between mobile users and wired networks. They are the key carriers for mobile communications. Wireless networks communicate by modulating radio waves or pulsing infrared lights. Compared to wired communication, wireless communication faces more obstacles because the surrounding environment interacts with the signal, blocks the signal path and introduces noise and echo. As a result, wireless communication is characterized by lower bandwidth, higher error rates, and more frequent spurious disconnection [FORM94]. There are various types of wireless networks: wide area cellular networks, wireless LANs, wireless PANs (personal area networks) and satellite networks. Each of which provide service over a variety of geographical coverage areas at various speeds and at a variety of price levels. Mobile users need to be able to access multiple networks in order to meet their needs [HILL96].

The available wireless technologies strongly dictate the kind of services that can be practically provided. In the following sections, we give a brief summary of different wireless technologies and services.

2.3.1 Wide Area Wireless Technologies and Services

-Cellular Technologies and Services

The three major cellular technologies are GSM, IS-136, and CDMA. In 1999, the primary cellular-based data services are Cellular Digital Packet Data (CDPD), circuit-switched data services for GSM networks, and circuit-switched data service for CDMA networks. All of these services offer speeds in the 9.6 Kbps to 14.4 Kbps range [RYSA99b]. The cellular industry delivers data services to users in two ways. One way is using a smart phone as a data device. With a microbrowser included in it, a user can view specially formatted Internet information through a smart phone. The other way is through wireless modems, supplied either in PC Card format or by using a cellphone with a cable connection to a computer.

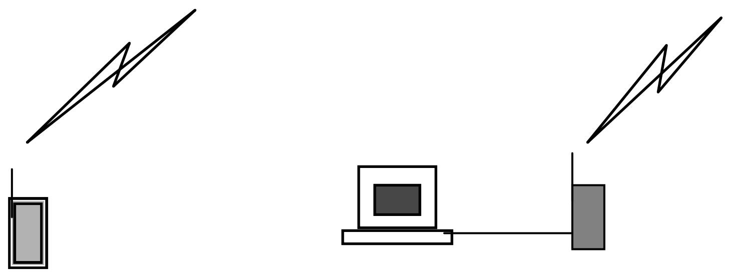


Figure 2.2. Smart phone versus phone connected to laptop

The current generation of cellular networks is referred to as second generation. New feature advancements to the current network are called 2.5G. Generally, 2.5G technologies have been developed for third generation (3G) networks, but they are applied incrementally to existing networks. All new technology networks are referred to as 3G networks. In the following we summarize each technology and its services.

GSM:

GSM is the most mature digital-cellular standard. It offered circuit-switched data services well in advance of other networks. A service in trail in 1999 is called high-speed circuit-switched data service (HSCSD). HSCSD combines two to four of the time slots (out of a total of 8 in each frame) to provide service from 28.8 Kbps to 56 Kbps. HSCSD is attractive to carriers because it requires minimal new infrastructure. Another service supported by GSM is General Packet Radio Service (GPRS), a 2.5G technology. GPRS can combine up to 8 (out of 8 available) time slots in each time interval for IP-based packet data speeds up to a maximum theoretical rate of 160 Kbps. The phase after GPRS is called Enhanced Data Rates for GSM Evolution (EDGE). EDGE introduces new methods at the physical layer, including a new form of modulation (8 PSK) and different ways of encoding data to protect against errors. EDGE will deliver data rates up to 500 Kbps using the same GPRS infrastructure. The 3G version of GSM is Wideband CDMA (WCDMA)

IS-136:

IS-136 supports circuit-switched data. Though developed initially for GSM, the IS-136 industry is trying to migrate EDGE to IS-136 networks. Deploying EDGE in IS-136 will require new radios in base stations to support the 200 kHz data channels. Deploying EDGE in IS-136 will allow users to roam between IS-136 and GSM networks around the world. IS-136 carriers might eventually use WCDMA, though a wideband TDMA (WTDMA) has been proposed [RYSA99b].

CDMA:

Currently CDMA carriers use circuit-switched technology operating at 14.4 Kbps. As with GSM, CDMA requires a handset that specifically supports data. Connect the phone to a laptop, and the phone operates just like a modem, enabling you to establish dial-up connections to the Internet. Another service for CDMA networks is called QuickNet Connect. By eliminating conventional modem connections, this service allows fast connections (of approximately five seconds) to the Internet. Today's CDMA service is based on the IS-95A standard. A refinement of this standard, IS-95B, allows up to eight channels to be combined for packet-data rates as high as 64 Kbps. Beyond IS-95B, CDMA evolves into 3G technology in a standard called CDMA2000 [RYSA99b].

Table 2.2 [RYSA99b] summarizes the technologies and services discussed above. The future trend is that all the technologies are slowly converging, provide ubiquitous coverage, and make the systems interoperate more readily.

Table 2-2. Summary of Cellular-Data Services

Core Technology	Service	Data Capability
GSM	Circuit-switched data based on the standard GSM 07.07	9.6 Kbps or 14.4 Kbps
	High-speed circuit-switched data (HSCSD)	28.8 to 56 Kbps service likely
	General Packet Radio Service (GPRS)	IP and X.25 communication over 14.4 Kbps
	Enhanced Data Rates for GSM Evolution (EDGE)	IP communication to 384 Kbps. Roaming with IS-136 networks possible.
	Wideband CDMA (WCDMA)	Similar to EDGE but adds 2Mbps indoor capability. Increased capacity for voice.
IS - 136	Circuit-switched data based on the standard IS-136	9.6 Kbps
	EDGE	IP communication up to 384 Kbps. Roaming with GSM networks possible.
	WCDMA or Wideband TDMA (WTDMA)	Similar to EDGE but adds 2Mbps indoor capability
	Circuit-switched data based on the standard IS-707	9.6 Kbps or 14.4 Kbps
	IS-95B	IP communication up to 64 Kbps
	CDMA2000 - 1XRTT	IP communication up to 144 Kbps
	CDMA2000 - 3XRTT	IP communication up to 384 Kbps outdoors and 2 Mbps indoors

2.3.2 High-Bandwidth Local Area Wireless Technologies

A wireless LAN (WLAN) is a flexible data communication system implemented as an extension to, or as an alternative for, a wired LAN within a building or campus. Using electromagnetic waves, WLANs transmit and receive data over the air, minimizing the need for wired connections. Manufacturers of wireless LANs have a range of technologies to choose from when designing a wireless LAN solution. Each technology comes with its own set of advantages and limitations [INTR]. Wireless LAN technologies include spread-spectrum, narrow-band technology, frequency-hopping spread spectrum, direct-sequence spread spectrum, and infrared technology. Wireless LANs operate at speeds up to a few Mbps, with radio ranges typically from 50 feet to over 1000 feet, depending on the specific product and the environment in which it operates. Many of the wireless LAN products can be interfaced directly to IEEE 802.3 (Ethernet) or 802.5 (token ring) wired LANs. A new standard IEEE 802.11 has been developed to allow interoperability between wireless LANs. Computers need to be equipped with a PC Card wireless LAN interface to access the wireless LAN. Software has been developing to allow users to seamlessly move between low-bandwidth CDPD networks and high-bandwidth Wireless LANs. Wireless LAN technology, typically conforming to the 802.11b standard, usually is found on notebook PCs.

2.3.3 Wireless PANs

Wireless PANs (personal area networks) are the wireless networks that can be installed in a small office or home within 5-15 meter distances. Two technologies being used for this purpose are IrDA that is based on line of sight requirement within two devices, usually a few feet apart, and BlueTooth which supports multipoint connection without line of sight requirement. The data speeds on Bluetooth range from 0.5 to 1.5 megabits per second (Mbps).

2.3.4 Satellite Networks

Even wireless wide area networks may not provide coverage in extremely remote areas, because service providers can not justify the economics of installing a base station everywhere. True universal coverage for these types of mobile users is possible only through wireless networks based on satellites. There are several GEO, LEO, and MEO satellite-based networks that transmit from different heights from the surface of the earth.

2.4 Mobile Software System

The previous sections are dedicated to mobile hardware systems. In this sections we discuss mobile software systems. A mobile software system contains the following components:

- Mobile operating systems for mobile devices.
- Application client software running on mobile computers
- Application server and/or database server software on workstations in wired networks
- Mobile middleware

The following figure displays the architecture of the software system for a mobile client [CHAN99].

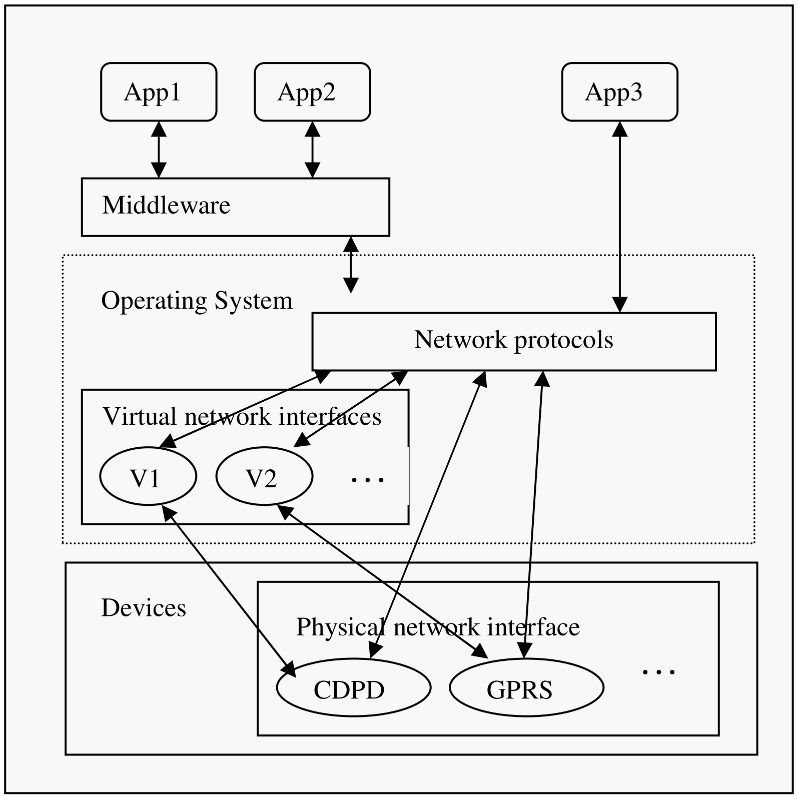


Figure 2.3. Mobile client architecture

The fundamental limitations of mobile hardware systems have important impact on the design of mobile software systems. The software system has to be small to meet the memory requirement; the algorithms have to be efficient to optimize the utilization of limited computing power and battery. In the following we discuss the operating systems, middleware, and applications respectively.

2.4.1 Mobile Operating Systems

The mobile operating system is responsible for managing physical resources, power consumption and communication interface. A mobile operating system is composed of the computer kernel, the power management facility, and the real-time kernel. The computer kernel is the entity that manages access to physical resources, such as CPU and disk space. The power management facility is responsible for reducing power consumption. The real-time kernel is responsible for managing the communication link [LAYN99].

The currently available mobile operating systems include Windows CE from the Microsoft Corporation, Palm OS from Palm, and Symbian EPOC from the Symbian LTD. These operating systems differ in memory requirements, the range of applications that can be supported, communications capability and interfaces.

2.4.2 Mobile Middleware

Mobile middleware is the layer of software that lies between the mobile applications and the underlying wireless networks. The middleware can be found at 3 different locations: the mobile client, the mobility gateway, and the information server. These middleware at different locations work together to shield users from the underlying wireless networks and the mobility of the users.

The range of services provided by a mobile middleware system to the application layer may include adaptivity and QoS, service management, TCP/IP functionality, disconnected operations, proxies and agents, communication resilience, and continuous data synchronization. By delegating the complex tasks of dealing with wireless communications to the mobile middleware layer, individual applications can focus mainly on their own application–specific logic, thus significantly reducing their complexity [CHAN99]. Mobile application developers need only to develop their applications according to a mobile API specification. With a standardized mobile API, the applications that comply to such an API will be able to run across different wireless networks. Interesting mobile and terminal aware applications can be written to make full use of the intelligence embedded within the mobile middleware.

Some tasks which are common to applications in a related area can be put in the middleware. For example, location information service and distance calculation can be included in middleware and serve various location-based applications.

Mobile middleware plays a central and brokerage role in the interaction between applications and the external wireless environments, as well as with the internal system resources. This factor makes the middleware one of the most interesting and challenging components of the mobile software systems [LAYN99].

2.4.3 Mobile Computing Applications

The application layer occupies the pinnacle of the mobile software system [LAYN99]. Like the other layers of mobile software systems, designing mobile applications has to take into account of the fundamental limitations of mobile systems. Mobile devices should not be considered general-purpose computers to run complex simulations and large applications due to their limited resources. Applications for mobile computers can be divided into the following five categories [KUNZ99]:

- Standalone applications such as games or utilities

- Personal productivity software (word processors, presentation software, calendars);
- Internet applications such as email, WWW browsers
- New location-aware applications such as tour planners and interactive guides.
- Ad-hoc network and groupware applications

The Internet applications constitute a very interesting and challenging category. This is the category we are interested in for this thesis. There are abundant Internet applications for desktop computers today. To introduce these applications into the mobile market place is an important ongoing trend with a high market potential. The applications getting the most attention are the ones that best take advantage of the mobility of wireless Internet devices, and which suffer least from the limitations of wireless platforms [HORI].

It's not surprising that e-mail is a top priority. E-mail is, in many ways, the perfect application for wireless devices, since technological constraints of small screen size and limited bandwidth do not limit e-mail as much as they do other applications. Another important mobile computing application is WWW mobile browser. It is the key part to extend the WWW services to mobile hosts. There are a number of browsers for handheld computers already. Since the cost of wireless services is high, general purpose web browsing might not be an attractive task for mobile users. Mobile browsers designed to provide a specific information service (e.g. financial information, daily banking, stock trading, weather information, ...) could be of more interest.

With the prospect that mobile devices may become the de facto standard for purchases, mobile e-commerce is a big business, and, as we might expect, it is another one of the top priorities for application developers. Finally, location-based services, for instance, providing users with information about restaurants they are close to is one of the most obvious advantages of the wireless Internet [HORI].

Developing applications for mobile systems is an area full of imaginations, challenges and with huge future market potential.

CHAPTER 3

RELATED WORK

-Improving Performance of Mobile Computing Applications

As discussed in the previous chapters, a mobile computing system has its intrinsic constraints. Mobile devices are resource poor and wireless networks have low bandwidth and are unstable, this causes mobile computing applications to suffer from bad performance and mobile users to suffer from long latency. A lot of research has been carried out in this area and various techniques have been proposed to deal with this issue. The effort of improving performance of mobile computing systems contains every aspect of mobile environment, from hardware aspect to software aspect, from proper network configuration to efficient application design.

In this thesis we concentrate on techniques to improve mobile computing performance at the application level. We research techniques to reduce user perceived latency and effective wireless bandwidth usage in mobile computing applications.

For mobile computing users, they always want fast access to networks and low cost. This means short user perceived latency and small bandwidth consumption. Research in recent years in mobile computing has proposed many techniques for optimizing data transfer over wireless links and utilizing mobile device resources. The commonly used techniques include caching, prefetching and compression. Each technique has its strength and weakness. Many of the techniques can be used in different combinations in different circumstances. These techniques often trade off one resource for another. For example, compression trades off computation for bandwidth, caching trades off memory for latency (and bandwidth), and prefetching trades off bandwidth for latency. The utility of these techniques thus depends on the service desired, the device, the network characteristics and the features of the application [CHAN99].

In the following sections, we review and discuss these techniques respectively. To keep the thesis consistence and make it easy to explain, we refer to all data objects in this chapter as files.

3.1 Caching

In a mobile computing environment, the bandwidth of the wireless link is very limited. This implies that each mobile client should minimize wireless communication to reduce the contention for the narrow bandwidth. Caching of frequently accessed data at the mobile clients has been shown to be a very useful and effective mechanism in handling this problem.

Caching improves the efficiency and reliability of data delivery over the network. A cache in the mobile client can serve a user's request quickly and reduce user perceived latency. When a user makes a request, the client checks the cache, if the required information is there, the client replies with it, otherwise, the client fetches it from the information server, replies to the user, and stores a copy in the cache as well. Other benefits of caching include energy savings and cost savings.

A caching strategy consists of the following components: cache size and replacement strategy. For mobile devices, since cache storage is limited, what to cache, when to cache, how long to cache, and how to replace the cached data have to be carefully considered. File size and data type impact the caching policy. Research in this area has resulted in many caching strategies, and they have been widely used in network applications, for example, in web browsers and mobile applications.

The efficiency of a caching strategy is measured by the following performance metric [BELL98]:

- The cache hit rate: the ratio of the number of files sent to users from the cache to the total number of files served from the cache and content servers on the Internet.
- The byte hit rate: the ratio of the average size of files sent to users from the cache to the average size of all the files served from the cache and content servers on the internet.

The byte-hit ratio gives more information of the network bandwidth. As files have different sizes, only recording the cache hit rate will not give a correct insight on how the strategy impacts the network bandwidth.

Cache consistency maintenance is another important issue to consider when applying caching in an application. Due to limitation of battery power, mobile computers may often be forced to operate in a sleep or even totally disconnected mode. As a result, the mobile computer may miss some cache invalidation reports broadcasted by a server, making the cached file out-dated or forcing it to discard the entire cache contents after waking up.

3.1.1 What to Cache and When to Cache

If there is space and the file is not dynamic, the file should always be cached. When space is insufficient, there are three simple strategies for deciding which files to cache: "cache all", which removes other files to make space; "threshold", the same as the previous strategy, but only caching files below a certain size; and "adaptive dynamic threshold," whereby the maximum file size threshold alters dynamically [REDD98].

3.1.2 Cache Replacement Schemes

File replacement strategies have a great impact on cache performance. The replacement process is tightly related to the cache size: if an infinite cache exists, there is no need for such a file removal process since all the files that enter the cache are kept for ever. In reality, the cached files have to be replaced because of a lack of memory space, and the choice of the next file to be removed is quite important [BELL98]. When deciding which file to discard, the key point is to discard the file that is most unlikely to be used again.

Factors that impact cache performance include: number of references to the files, the file size, the file time-to-live, and file retrieval time. For example, consider the factor of file size. For all caches, an upper limit is defined for the size of the caching area. If the cache is full when it receives a request to store a large file, what is the best strategy to maximize caching benefit? Is it more sensible to replace a single large file than several smaller ones [BAEN97]. It becomes more and more clear that an appropriate algorithm for selecting the files should consider several factors. However, the appropriate algorithm to combine these factors has not been defined yet.

The file replacement strategy can be split in two phases: first, the documents are sorted in order to determine the removal order, second, one or more documents are removed from the removal list. Cache replacement strategies can be divided into two groups: on-demand replacement strategies and periodic replacement strategies. In an on-demand replacement strategy, the removal process is started once the cache is full or the remaining amount of memory is not enough to store the new incoming file. In a periodic replacement strategy, files are removed after a certain amount of time. The possible disadvantage of the on-demand replacement strategy is that if a cache is nearly full, then running the removal policy only on-demand will invoke the removal policy on nearly all document requests, if removal is time consuming, it might generate a significant overhead. On the other hand, periodic removal reduces the hit-ratio because files are removed earlier than required [Bell98]. In the following we study different cache replacement strategies.

- Random replacement

A random replacement simply discards a randomly chosen file.

- The least recently used (LRU): The simplest and most common used cache management approach, which removes the least recently accessed files until there is sufficient space for the new file. This approach not only stores data also the time that each file was last referenced. LRU is widely used as a replacement policy. It is well understood and shown to perform well in a wide variety of workloads.

- The least frequently used (LFU): LFU keeps track of the number of times a file is used and replaces the least frequently used file to make room for an incoming file. LFU is more complex to implement than LRU. In LRU, a referenced file is just placed at the head of removal list. In LFU, the referenced file needs to be placed in a sorted (by frequency of use) list and this in general is more complex. One modified version of LFU is LFU-aging. LFU-aging allows an incoming file to replace only blocks with a frequency count of 1.

- First-in-first-out (FIFO): discards the file that is the oldest in the cache. It is easily implemented as a circular buffer.

- Largest-file-first (LFF): discards the largest file in the cache.

There are a lot of hybrid strategies and generalizations of the above caching strategies. Some generalizations of LRU attempt to make it more sensitive to the variability in file size and retrival delays, like GreedyDual algorithm [YOUN91]] and Greedy-Dual-Size (GDS) [CAO97, IRAN97]. Some generalizations of LRU attempt to incorporate access frequency information into LRU, like LRU-K [NEIL93] and LNC-W3 [SCHE97, SHIM99].

All the above methods only consider a single factor. Research has shown that a single or a combination of several parameters in the file replacement strategy is not enough to achieve higher performance. New methods have been proposed in [BELL98]. In these methods, each document is assigned a priority according to which the removal process will select the next document to be replaced. The priority is computed according to a certain mathematical model. In each model, the different factors are assigned a weight which is calculated using weight optimization methods or simulation techniques [BELL98].

3.1.3 Cache Consistency Maintenance

For caching to be useful, cache consistency must be maintained, that is, cached copies should be updated when the originals change [CAO98]. Cache consistency policies can be divided into two groups [CAO98]: weak consistency and strong consistency. Weak consistency is a consistency model in which a stale document might be returned to the user. Strong consistency is the consistency model in which, after a write completes, no stale copy of the modified document will ever be returned to the user.

Weak Consistency Approaches:

Existing Web caches mostly provide weak consistency. Weak consistency mechanisms include TTL (Time-To-Live) and client polling. In TTL, a client considers a cached copy up-to-date if its time-to-live has not expired. In client polling, a client periodically contacts web servers to verify the freshness of cached copies.

The difficulty with the TTL approach is that it is hard to assign an appropriate time-to-live for a file. If the value is too small, the server will be burdened with many " if-modified-since" messages, even when the file is not changed. If the value is too large, the probability that the user will see a stale copy of the file significantly increases. Similar difficulties exist with the client-polling approach in deciding when to send " if-modified-since" request. Adaptive TTL, a generalization of TTL, is a widely used weak consistency approach. The adaptive TTL approach handles the problem by adjusting a file's time-to-live based on observations of its lifetime.

Strong Consistency Approaches:

Strong consistency approaches include invalidation and polling-every-time. In the invalidation approach, the content server keeps track of all the client sites that cache a document and, when the document is changed, sends invalidation message to the clients. A write is considered complete when invalidation messages reach all of the relevant clients. In the polling-every-time approach, every time the user requests a document and there is a cached copy, the cache first contacts the server to validate the cached copy, then returns the copy. In this approach, a write is complete when the modification is registered in the server's file system.

Weak consistency protocols save network bandwidth mostly at the expense of returning stale documents to the users, and the comparison of invalidation and polling-every-time depends on the relative frequency of document requests and modifications [CAO98].

3.2 Prefetching

Prefetching is a technique which allows a client to download information in advance, ready for the user to use later. Prefetching tries to reduce the perceived latency by look-ahead. If a user can anticipate what files he will fetch in the future, it is possible to preload these files to the client before they are requested.

The idea of prefetching stems from the fact that, after retrieving some information (webpage, file, etc), the user usually spends some time viewing and processing the information. During this period, the connection link is idle, if we can take advantage of this phenomenon by prefetching some information that will likely be accessed soon to the local machine, there will be no transmission delay when the user actually requests them [JIANG98a]. The perceived latency can then be reduced to that for fetching information from local disk or file servers on local nets.

Prefetching is only effective when future access pattern can be determined. The key challenge in prefetching is to determine "what to prefetch, when to prefetch, and how much to prefetch" so that performance will be enhanced. The difficulty of realizing efficient prefetching lies in the fact that it is impossible to predict exactly what a user is going to need and hence some prefetched files may never be used [JIANG98a]. In mobile network environments, link bandwidth is a very scarce and high cost resource, prefetching information based on inacurate prediction which are not used by the user will waste precious bandwidth. For the overall system, applying a prefetching scheme may increase the network load, because prefetching can cause downloading files that are not used. Therefore it degrades the whole system performance. So, for prefetching in network applications, there is tradeoff between user perceived latency and bandwidth. For applications with an obvious access pattern, a prefetching scheme is suitable. For applications with an arbitrary access pattern, a prefetching scheme is not suitable.

The prefetch problem has two aspects: the first one is how to estimate the probability of each file being accessed in the near future. In general, this probability changes with time as the user issues new requests. The second aspect is how to determine which files to prefetch such that the overall system performance can be optimized relative to some criteria [JIANG98b].

The performance of a prefetch scheme is measured by the following criteria:

- Hit rate: refers to the percentage of user requests satisfied by the prefetched files. Generally, the higher the hit rate, the more reduction in user-perceived latency is achieved.

- Successful prediction rate: the probability of a prefetched file being used eventually. A high successful prediction rate indicates that less bandwidth is wasted due to prefetching unused files.

Information on user's future access may be derived from server's access statistics, client's configurations, application's nature, user's personal preference, and user's planing tools.

In the following we review various existing prefetch techniques. These techniques are mainly targeted for Internet web access, but since the World Wide Web is a very common and typical network application, these techniques can be applied to other network and mobile computing applications based on the features of the techniques and the applications.

3.2.1 Statistical Prefetching vs. Deterministic Prefetching

In [WANG96], statistical and deterministic prefetching schemes for Internet Web access are discussed. In a statistical scheme, the interdependence of web page accesses are calculated periodically based on the most recent access logs, web pages with interdependencies higher than a certain threshold are grouped for prefetching. Statistical prefetching can potentially have wide applications as the process can be easily automated. However, by its speculative nature, some bandwidth will be wasted, thus it can increase total bandwidth consumption. There is a general tradeoff between bandwidth and latency here. If we reduce the threshold for statistical prefetching, the latency may improve, but at the price of increased bandwidth consumption. Unfortunately, bandwidth is still a scarce resource in most networks, particularly over wireless links and long distance paths. Thus statistical prefetching has to be used with great care to avoid bandwidth waste. Note that, although the perceived delay for prefetched files are very low, the retrieving delay for non-prefetched files may actually increase as a result of the extra traffic load caused by prefetching. When traffic is heavy, aggressive prefetching, such as "get all links", may actually increase the average latency of all accesses. It is concluded in [WANG96] that, unless the traffic is very light or the prefetching efficiency is very high, statistical prefetching may not necessarily reduce perceived latency.

In a deterministic scheme, prefetching is configured statically by the users as part of their personalized user interface, or even by page designers as part of the content design. Deterministic prefetching is the most conservative types as it often has little or no bandwidth overhead, although its scope of use is limited. Nevertheless, when users know what needs to be prefetched, it can reduce perceived latency, and to some extent, even ease congestion at very little cost. In [WANG96], some potential uses of deterministic client-initiated prefetching are discussed. The following are some of the examples:

- Batch Prefetching: Web pages that are read on a regular basis, such as on-line newspapers, weekly work reports, can be prefetched in a batch mode during the less busy period.

- Start-up prefetching: when a browser is started, a set of pages users need to look at that day may be prefetched in the background.

- Pipelining with prefetching: for some information services where users can easily specify the sequence of pages to be viewed, such as on-line newspapers, stock market prices, and headline tracking services, we can potentially pipeline the operations by fetching the next page while the user is looking at the current one.

3.2.2 URL Graph Based Prefetching vs. Context-Specific Prefetching

Prefetching algorithms used in today's commercial products are either blind or user-guided. A blind prefetching technique does not keep track of any user web access history, it prefetches document based solely on static URL relationships in a hypertext document. User-guided prefetching techniques are based on user-provided preference URL links [IBRA00].

Research in prefetching is targeted at intelligent predictors based on user access patterns. A popular approach is to represent the access pattern as a URL graph. The URL graph based approaches are demonstrated effective for documents that are often accessed in history. However, they are unable to pre-retrieve those documents whose URLs are never touched before. A new approach, context-specific prefetching, is proposed in

[IBRA00], which overcomes this limitation. It relies on key words in anchor texts of URLs to characterize user access patterns and on neural networks over the keyword set to predict future requests.

3.2.3 Real-time Online Adaptive Prefetching vs. Off-line Prefetching

In [JIANG98a], a general prefetching scheme is describe for real-time online web access. We refer to it as an adaptive network prefetch scheme. The adaptive network prefetch scheme comprises a prediction module and a threshold module. The prediction module computes the access probability, which is defined as the conditional probability of a file being requested by the user in the near future, given the file currently being used. The threshold module computes a prefetch threshold for the information server based on network and server conditions as well as the costs of time and bandwidth to the user, such that the average delay is guaranteed to be reduced by prefetching files as long as their access probability is greater than its server's prefetch threshold. The prefetch threshold is a function of current system condition and certain cost parameters.

The prefetching scheme works in the following way: Basically, whenever a new file is requested, the prediction module updates the local access history, if needed, and computes the access probability of each file somehow related to the current file. At the same time, the threshold module computes the server's prefetch threshold. Finally all the files with access probability greater than the server's threshold are prefetched.

This prefetch scheme is a general one that may be applied to almost any network application to decide what information to prefetch. For specific network information access application, the problem of applying this prefetching scheme is to decide the prediction algorithm and the algorithm to compute the server threshold. Some examples of the prediction algorithm and threshold algorithm are given in [JIANG98a].

[JIANG98a] also described a prefetching scheme to deal with disconnectivity of mobile clients. This scheme allows users to prefetch a group of files together for the user who is about to disconnect from the network. In this case, the prefetching threshold is not applicable. The prediction module is still used. At first the user specifies an upper bound for the bandwidth cost or the amount of data to be downloaded. The user also specifies some initial files. The system starts to download files. First initial files are downloaded, after downloading each file, access probabilities of other files are calculated, and the file with the highest access probability is downloaded. This process continues until the total bandwidth cost or the total amount of data downloaded exceeds the user specified limit. More sophisticated functions can be added to select files more intelligently [JIANG98a].

Research has shown that prefetching is a good approach to reduce latency for network applications. However, it must be implemented in such a way that the overall system performance can be improved, and the tradeoff between saving user's time and bandwidth usage must be considered.

3.3 Compression

Compression is the reduction in the size of data in order to save space or transmission time. Compression is performed by a program that uses a formula or algorithm to determine how to compress or decompress data.

In network applications that involve massive data transmission, minimizing communication cost is a vital and challenging task. Data compression makes use of relatively abundant CPU computation power to minimize the amount of data to transmit, equivalently reducing the bandwidth requirement. Since data compression is virtually suitable to all types of platforms and communication media, it has been long existent and widely put into practice [CHIO00].

In mobile computing systems, transmitting data over wireless links consumes the limited and precious wireless bandwidth and battery power. Incorporating data compression during transmission can be a feasible approach to save bandwidth and battery power. However, blind, or unconditional compression may result in a waste of CPU power and even degrade the overall system performance. When and under what circumstances to apply data compression has to be carefully considered.

Applying compressing in communication is effective or “good” if the time of transmitting compressed data, plus the compressing/decompressing overhead, is less than the time of transmitting uncompressed data. A good data compression scheme should dynamically trade network bandwidth with CPU computation power or vice versa, depending on circumstances [CHIO00].

Data compression techniques can be divided into two categories: lossless and lossy compression [CHIO00]. Lossy compression is usually applied to graphical and audio objects, and lossless compression is applied to text and binary objects. They are summarized below.

3.3.1 Lossless Compression

With lossless compression, every single bit of data that was originally in the file remains after the file is uncompressed. All of the information is completely restored. This is generally the technique of choice for text or spreadsheet files, where losing words or financial data could pose a problem. The Graphics Interchange File (GIF) is an image format used on the Web that provides lossless compression.

Lossless compression can be further divided into three subcategories: dictionary-based, statistical, and prediction-based coding [CHIO00].

- Dictionary-based Compression

Dictionary based compression systems operate by replacing groups of symbols in the input text with fixed length codes. A well-known example of a dictionary technique is LZW data compression. LZW operates by replacing strings of essentially unlimited length with codes that usually range in size from 9 to 16 bits.

- Statistical Compression

Statistical methods of data compression take a completely different approach. They operate by encoding symbols one at a time. The symbols are encoded into variable length output codes. The length of the output code varies based on the probability or frequency of the symbol. Low probability symbols are encoded using many bits, and high probability symbols are encoded using fewer bits.

The compression ratio of the statistical coding schemes, such Huffman coding and arithmetic coding are affected by the degree of skewness in the message sequences. Most data however only exhibit small skewness, thus most statistical coding schemes do not perform well [BASS90].

- Prediction-based Compression

Prediction-based techniques are represented by Markov models.

The dictionary-based coding schemes run faster and achieve better compression ratio than statistical coding ones in general. Markov algorithms yield the best compression ratio, but are much slower and need more storage space than dictionary–based algorithms. Therefore dictionary-based techniques are preferred under most circumstances [CHIO00].

3.3.2 Lossy Compression

In lossy compression, information is thrown away during compression, so that the original data cannot be recovered by decompression. Instead, decompression produces an approximation to the original data, with the level of approximation dependent on the compression ratio. The loss of data makes it possible to have compression ratios that are typically an order of magnitude greater than lossless compression ratios. Lossy compression is generally used for video and sound, where a certain amount of information loss will not be detected by most users. The JPEG image file, commonly used for photographs and other complex still images on the Web, is an example of lossy compression. Using JPEG compression, the creator can decide how much loss to introduce and make a trade-off between file size and image quality.

Data compression has been a topic of long history and with many applications. Methods of data compression have been studied for almost four decades [LELE87]. Numerous methods and algorithm are exists. Due to the constraints of time and space of this thesis, we will not going to the details of these algorithms. We just gave a brief introduction of the concepts and theory of this area.

3.4 Summary

In this chapter, we reviewed the existing techniques in improving performance of network applications with the aim to utilize them in mobile computing environments. Numerous research and results exist. We are not intending to cover everything. We concentrate on caching, prefetching and compression techniques. These are important areas with extensive research. Every technique has its strength and weakness. Many of the techniques can be used in different combinations in different circumstances. These techniques often trade off one resource for another. The utilization of these techniques thus depends on the service desired, the device, the network characteristics and the features of the application.

CHAPTER 4

A MOBILE COMPUTING APPLICATION - MOBILE MAP

In this chapter, we develop a mobile computing application - mobile map. The first benefit of this work is that this is an interesting mobile computing application. The second benefit is that it can be used as a base to study the effect of various prefetching and caching techniques on the performance of the application.

This chapter is organized as follows. Section 4.1 gives a description of the application. Section 4.2 covers the application software design and architecture. The features of the application are analyzed. The major components and their cooperation relationship are discussed. Section 4.3 describes the application development environment, implementation process and main issues. Section 4.4 summarizes the results.

4.1 Application Description

The Mobile Map is a client/server network application running on a user's handheld computer and an Internet map server. When a user is on the move (traveling, driving, exploring the nature), with a palm computer at hand and with the palm computer connected to the Internet map server via a wireless network, also with a GPS device integrated in the Palm computer, he will always see where he is and where he is heading to. He will never worry about getting lost. The palm computer receives the user's geographic location information through the GPS system. It downloads the map segment around the user's location from the Internet map server, and displays the user's moving trace and the map segment on the palm computer screen. The user can also choose to view the map at different resolution levels. Whenever the user changes the map resolution level or moves out of the current map segment, the client detects the change and automatically downloads the next map segment and displays the user trace and the new map segment.

4.2 Application Analysis and Design

Object oriented analysis and design approaches are used to develop this application.

4.2.1 Client/Server Model

As described in Section 4.1, the Mobile Map is a client/server network application. The client resides in a mobile computer (handheld, palm…) featuring small memory, small storage, small display, slow performance, slow network bandwidth, less functionality, and less user friendly input method. The server resides in a large computer or workstation in the Internet that has large memory, large storage, and fast processing ability. The following figure shows the Mobile Map client/server model.

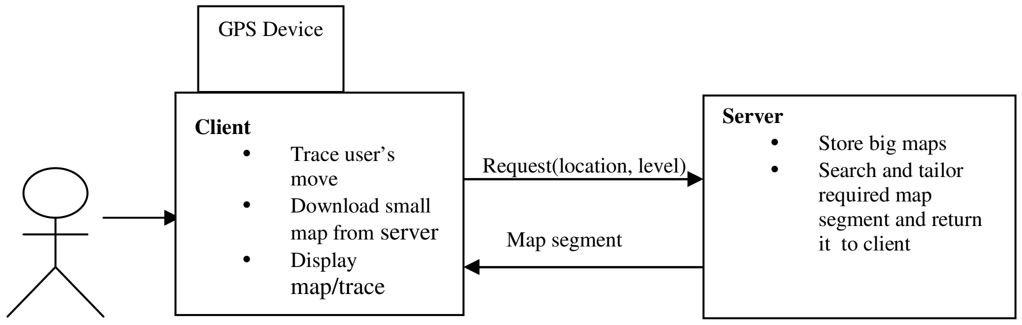


Figure 4.1. Mobile Map Client/Server Model

In this research, we concentrate on a single server and a single client model. Big maps of different resolution levels are stored in the map server. The server listens to client requests, searches and processes the required map segment, and returns it to the client. The client traces a user's move with the assistance of the GPS system, and requests relevant map segments from the server. Upon receiving the requested map segment, the client displays the map segment and user trace on the hand held computer screen. Whenever the user moves out of the current map segment or the user requests a new resolution level, the client issues a new request for a new map segment.

4.2.2 Main Components and Application Architecture

The Figure 4.2. shows the main components and architecture of the application.

(a) Basic Transmission units: Request and MapUnit are the two basic data units transmitted between client and server. A request contains user location (latitude, longitude), map resolution level (0 - 4) and user moving direction (0-NE; 1-NW; 2-SW;

3-SE). A MapUnit contains a small map segment and the coordinates (latitude and longitude) of the top-left point and bottom-right point of the map segment.

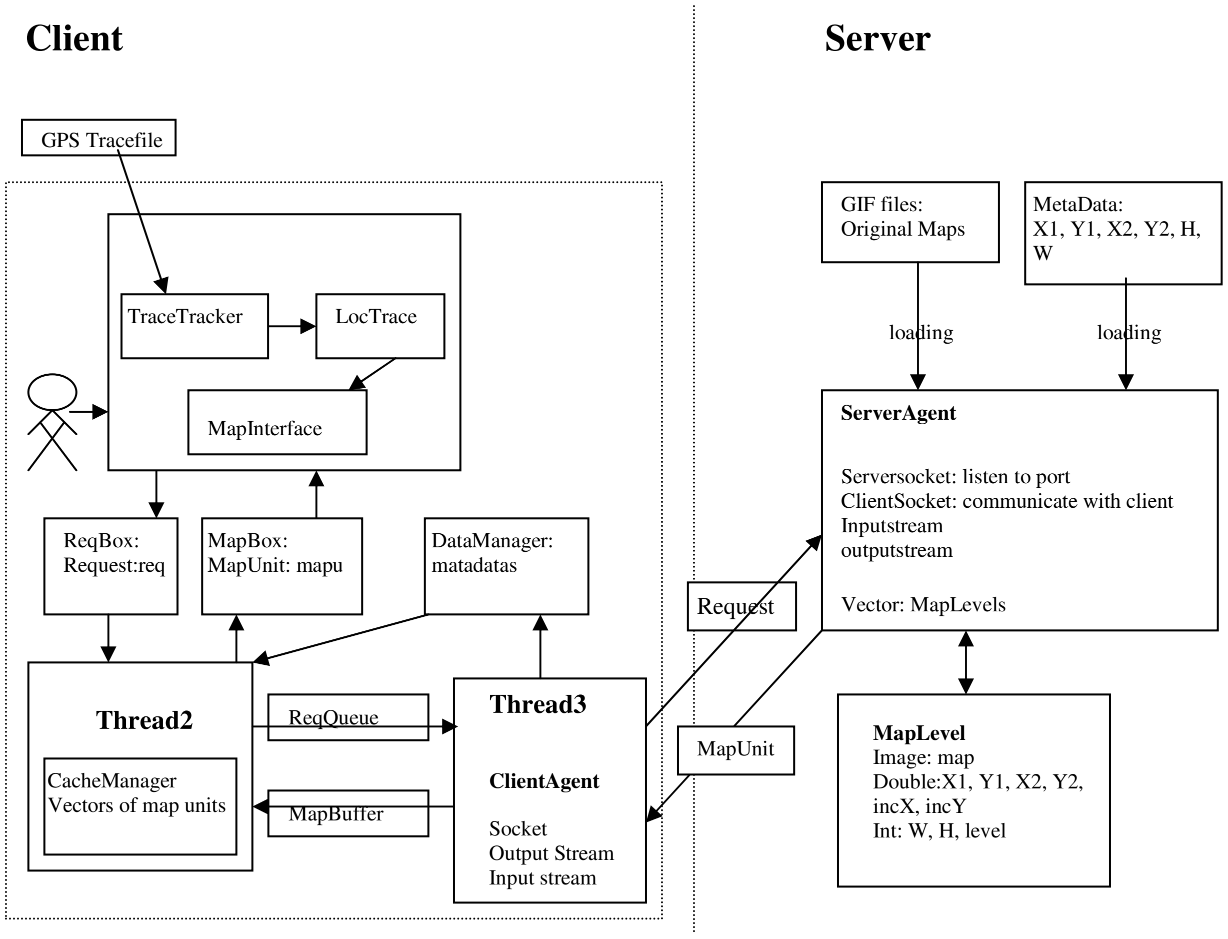


Figure 4.2. Main Components and Architecture of the Mobile Map

Main components at the client side:

(b) MapInterface (see Figure 4.3): This is a GUI user interface that resides in the client. It contains a canvas for displaying the map and user trace and a panel containing three buttons: ZoomIn, ZoomOut, and Exit, which allow the user to view a map at different resolution levels or exit the application at any time. The display screen size is 160 pixels*160 pixels. The displayed map size is 160 pixels*150 pixels.

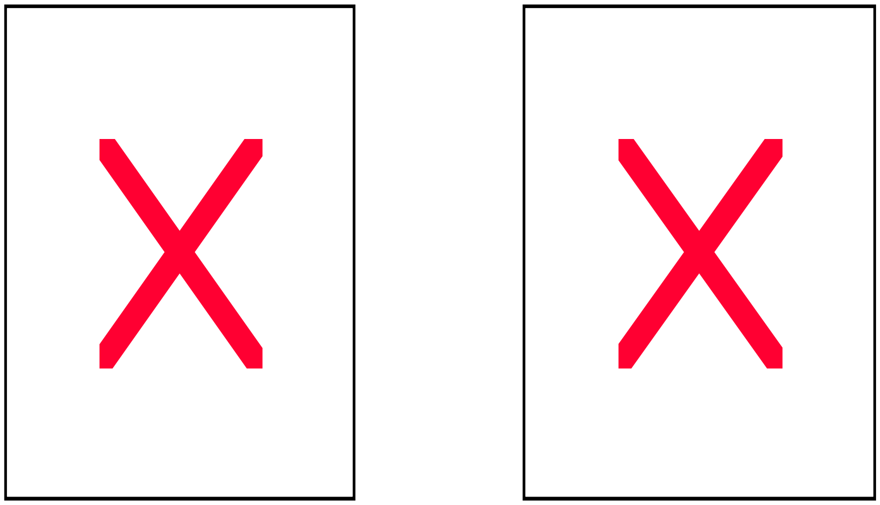


Figure 4.3. Screenshots of the Mobile Map Application

(c) TraceTracker: It traces user movement (trace of locations) and returns the user's next location.

(d) CacheManager: It is the storehouse of map segments at the client side. Upon receiving a new request from MapInterface, it searches its map cache for the requested map segment, if found, send it to the MapInterface, otherwise, pass the request to the client agent to retrieve it from the server on the Internet.

(e) ClientAgent: Its responsibility is to communicate with the server, pass requests to the server, and receive requested map segments from the server.

Main components at the server side:

(f) ServerAgent: It is responsible for the communication with the ClientAgent. When receiving a request from the ClientAgent, it passes the request to the server to search for the required map segment and sends it to the ClientAgent.

(g) Server: It is responsible for loading and storing all initial big maps. When receiving a request from the server agent, it searches and fetches the required map segment and passes it to the ServerAgent.

4.2.3 MultiThreaded Programming

Three threads are implemented in the application. Thread1 traces the user's movement and displays the user trace and map on the palm screen. It also informs Thread2 of new requests whenever the user moves out of the current map segment or the user requests a new map resolution level.

Thread2 manages the map cache, responds to Thread1's requests and stores map segments received from Thread3. It listens to Thread1, whenever a new request comes, it searches the map cache for the required map segment, if found, it returns the map segment to Thread1, if not found, it passes the request to Thread3. It collects map segments received from Thread3 and puts them in the map cache. Caching and prefetching schemes are implemented in Thread2.

Thread3 is responsible for communication with the server, fetching required map segments from the server and passing them to Thread2.

Thread1 and Thread2 communicate through a request box and a map box. Requests pass from Thread1 to Thread2 through the request box. Map segments pass from Thread2 to Thread1 through the map box. Thread2 and Thread3 communicate through a request queue and a map buffer. Requests missed from the cache and generated by the prefetching are put in the request queue by Thread2. Thread3 fetches requests from the request queue and retrieves map segments from the server. The received map segments are put in the map buffer. Thread2 collects the map segments in the map buffer.

The following figure shows the three-threaded architecture of the application.

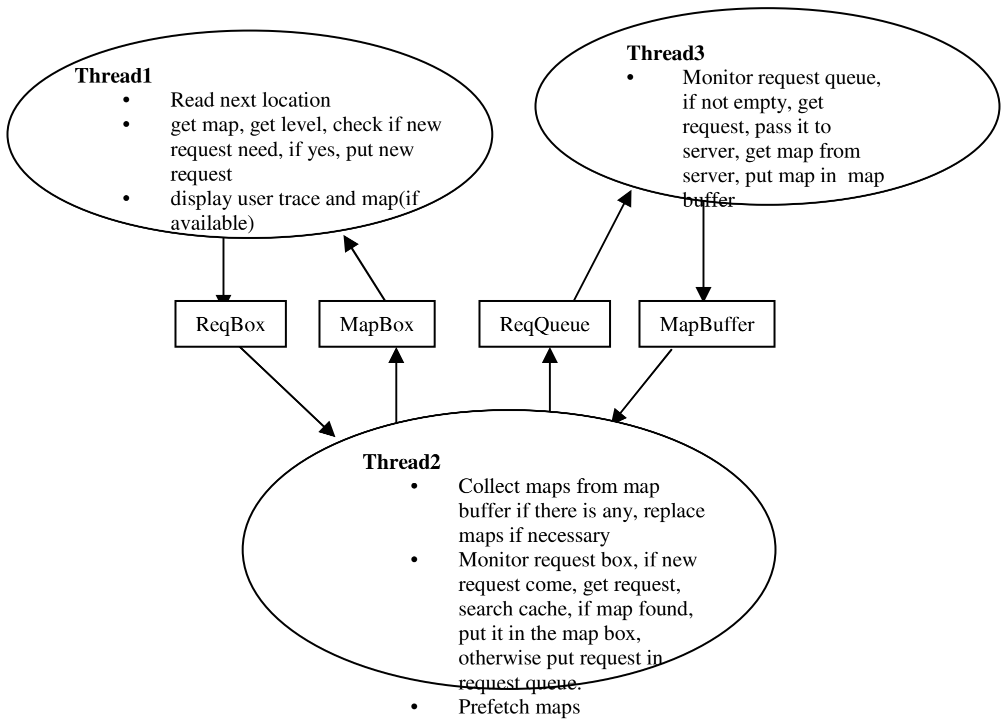

Thread1
• Read next location
• get map, get level, check if new request need, if yes, put new request
• display user trace and map(if available)
Thread3
• Monitor request queue, if not empty, get request, pass it to server, get map from server, put map in map buffer
ReqBox
MapBox
ReqQueue
MapBuffer
Thread2
• Collect maps from map buffer if there is any, replace maps if necessary
• Monitor request box, if new request come, get request, search cache, if map found, put it in the map box, otherwise put request in request queue.
• Prefetch maps

Figure 4.4. Three-threaded Architecture of the Application

This three-thread implementation is to insure the real time feature of the application. Thread1 continuously tracks the user's move and displays the user trace and map (if available) while Thread2 and Thread3 are searching and retrieving required map segments.

4.2.4 Operations Sequence

Figure 4.5 shows the operations sequence of the application.

Server Algorithm:

```
Load in maps
Establish socket connection between ServerAgent and  ClientAgent
Send map meta data to client
While true
{
    Get request from client
    If (request=End) exit;
    Else
    {
        Search map segment
        Send map segment to client
        End if get stop signal from client
    }
}//end of while
```

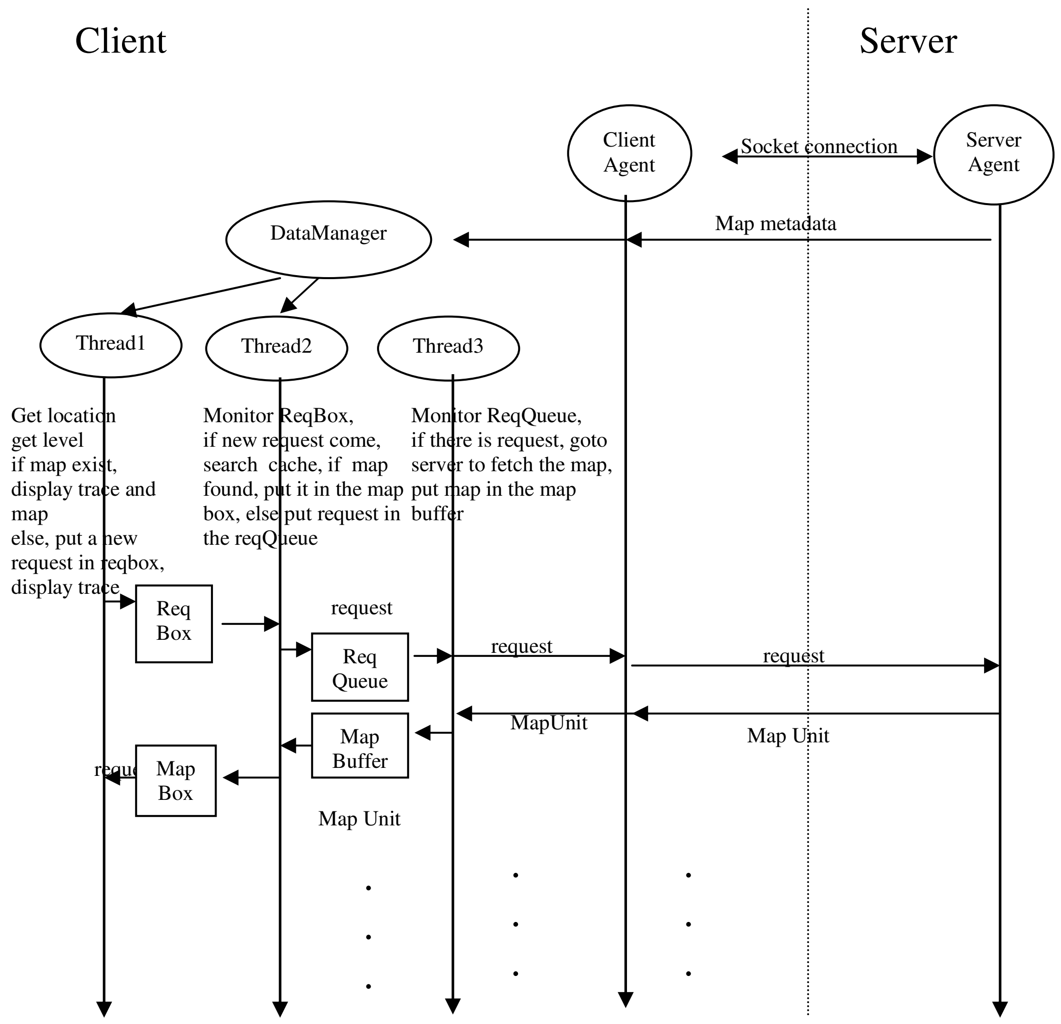

Client
Server
Client Agent
Socket connection
Server Agent
Map metadata
DataManager
Thread1
Thread2
Thread3
Get location get level if map exist, display trace and map else, put a new request in reqbox, display trace
Monitor ReqBox, if new request come, search cache, if map found, put it in the map box, else put request in the reqQueue
Monitor ReqQueue, if there is request, goto server to fetch the map, put map in the map buffer
Req Box
request
Req Queue
request
request
MapUnit
Map Unit
Map Buffer
Map Box
Map Unit

Figure 4.5. Operations Sequence of the Mobile Map Application

Client Algorithm:

Establish socket connection between ClientAgent and ServerAgent

Receive map meta data from server

```
    Thread1 start running
    Thread2 start running
    Thread3 start running
```

Thread1 Algorithm:

```
    get location;  //from log data
    update logtrace; //user trace
    get level; //from user input
    get direction; //from loctrace
    form a new request and put it in ReqBox;
    get map segment frame; //from DataManager
    display trace;
    while( true)
    {
        get location  //from log data)
        if (end of file ) put End request in ReqBox, exit;
        update logtrace; //user trace
        get level; //from user input)
        get direction; //from loctrace
        form a new request;
        if request matches current map segment frame
        {
            check MapBox
```

```
                if required map arrived, display map and trace
                else
                display trace
        }
        else //a new request
        put new request in the ReqBox.
}//end of while
```

Thread2 Algorithm:

```
    Get the first request from ReqBox;
    Put the request in ReqQueue;
    Execute prefetching;
    while( true)
    {
        Collect map segments from MapBuffer if there is any;
        If cache full, replace map;
        check ReqBox
        if (no new request)
        {
            check if current request has been met
             if not
            {
            search cache,
                if required map found, put it in MapBox.
                }
        }
        else // new request arrive
        {
                if (request=end), put End request in ReqQueue, exit;
                else
                {
                        clear ReqQueue;
                        search cache for required map segment,
                        if found, put it in MapBox;
                        else put request in ReqQueue.
                        Execute prefetching;
                }
        }
}//end of while
```

Thread3 Algorithm:

```
while( true)
{
        if there is request in the ReqQueue, get request
        if (request==end)
        send End request to server, exit;
        else
        {
                send request to server
                receive map from server
                put map unit in MapBuffer
        }
    }//end of while
```

4.3 Mobile Map Development Environment and Implementation

The Mobile Map server runs in a PC or workstation and is developed using Java JDK1.3. The Mobile Map client which runs in a palm computer is developed in J2ME CLDC (Connected Limited Device Configuration) Palm KVM Environment.

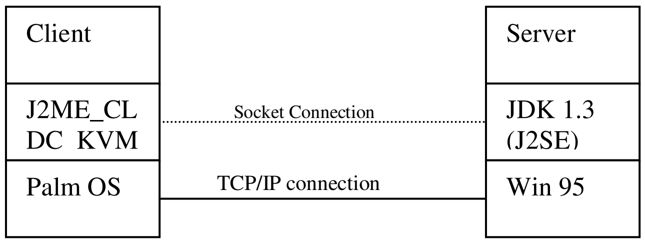


Figure 4.6. Mobile Map Development Environment

Sun's Java technology has three editions: Java 2 Enterprise Edition, (J2EE), Java 2 Standard Edition (J2SE) and Java 2 Micro Edition (J2ME). Each Java edition defines a set of technology and tools that can be used with a particular product.

- Java virtual machines
- Libraries and APIs specialized for each kind of computing, and
- Tools for deployment and device configuration

Java JDK1.3 belongs to the J2SE family. The J2ME is a new edition of the Java 2 platform targeted at consumer electronics and embedded devices. It is composed of a virtual machine and a minimal set of core libraries, and allows for extending the capabilities of the minimal configuration by adding additional libraries within the context of an industry-defined profile.

At a high level, J2ME is currently targeted at two broad categories of products:

- Shared, fixed, connected information devices, this category is represented by the grouping labeled CDC (Connected Device Configuration).
- Personal, mobile, connected information devices. This category is represented by the grouping labeled CLDC (Connected, Limited Devices Configuration)

J2ME Connected Device Limited Configuration specifies the core libraries and virtual machine features for J2ME implementation on highly constrained devices. KVM is a Java virtual machine that is included in the CLDC reference implementation. The following table shows the major difference between Java SE PC and Java ME (Palms).

Table 4-1. Difference between Java SE PC and Java ME (Palms)

	Java SE	Java ME_CLDC
Hardware Type	PC, Workstation	Handheld device, PDA
CPU Speed	100+MHz	16-20 MHz (slow)
Program Memory Size	16+MB	128-256 KB
Storage Size	1+GB	2-8 MB
Display Size	640x480 pixels+	160x160 pixels
Internet Connection Speed	56.6Kbps	19.2 - 33.6 Kbps for regular modems 14.4 Kbps for some wireless modems
Java Class Library	Full	subset
GUI Class Library	AWT, Swing	KAWT
Internet Connection Class Library	HTTP, Socket, Datagram	Socket
Math Class Library	Yes	N/A
Float & Double Data Types	Supported	N/A

A typical KVM device has "small memory", "small storage", "small display", "slow performance", "slow network bandwidth", "less functionality", and "less user friendly input method". To overcome these difficulties, the following issues have been considered when implementing the mobile map application.

- Converting Doubles to Integers

  Float and Double data types and mathematical functions are not supported in KVM. But in the mobile map application, user location (latitude and longitude) and map data (coordinates of left-top point and right-bottom point) are represented in the double data type. Since the client cannot process double numbers, the solution for this is to convert doubles into integers in the server before these numbers are sent to the client. For example, if user location is (75.67408, 45.39998), it will be converted to (7567408, 4539998). All other calculations are adjusted accordingly.

- Efficient User Interface

  The GUI is an essential component of the mobile map application. It is used to display the map and user trace on the palm screen and interact with the user to allow him/her to view maps at different resolution levels. Since a palm computer has a pretty small display screen, when we design the GUI for the mobile map application, we abide by the principle to keep it simple and clear. The following figure shows the GUI of the mobile map system. It contains a panel and a canvas. The panel has three buttons to allow a user to change map resolution levels or to exit the application. The canvas is the place to display map and user trace.

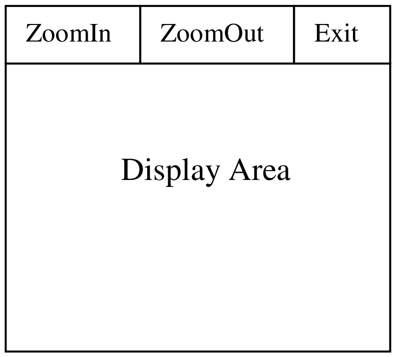


Figure 4.7. GUI of the Mobile Map System

The GUI of the mobile map application is implemented in KAWT that is a simplified version of AWT for the KVM.

- Transmitting Maps Through Socket Connection

Transmitting user requests and maps between client and server is the basic task of the application. The client/server communication channel is established by using the following Java classes: ServerSocket, Socket, InputStream and OutputStream. Due to the KAWT limitation, the I/O streams for the client are constraint to InputStream and OutputStream, which are used to read bytes from the socket and to write bytes to the socket. Any data (Integer, in this application), before it is sent into the socket, is converted into a stream of bytes. At the receiving end, the received byte stream is converted back to its original data. A map segment, before it is transmitted from server to client, is encoded into a stream of bytes of GIF format, then sent to the socket connection.

- Handling Images in Java KVM

At the server side, original maps are stored on the disk as GIF files. In this application, we have five big maps for the same area but five different resolution levels. When the server start running, the five maps are loaded into the application using the Java toolkit function:

Image Toolkit.getDefaultToolkit().getImage("filename.gif")

They are stored in the server as Image objects. When the server receives a new request from the client, the server finds the map at the requested resolution level, filters out the required map segment, uses a method gifEncoder (found on the Internet, compression function is included in this method) to convert the Image object of the map segment into an array of bytes of GIF format, then sends the byte stream to the client through the client/server socket connection.

At the client side, the KAWT classes: ToolkitImpl, GraphicsImpl, ImageImpl and their methods are used to handle map segments. The received byte stream of GIF data of a map segment is converted into an Image object using the following method in the ToolkitImpl class:

Image Toolkit.getDefaultToolkit ().createImage (byte [ ] data)

The map segment (Image object) and user trace are displayed on the screen using methods in the GraphicsImpl class.

4.4 Summary

In this chapter, we analyzed the characteristics and requirements of the application, described the design and development process of the application. OOD, OOP and Java technologies are used to develop this application. The main components, program architecture, operations sequence and algorithms are described. The mobile map application is a client server network application. The client which runs on a palm computer is developed in Java KVM. Due to the constraints of the palm computer and Java KVM, some particular issues are addressed (GUI design, socket communication, data conversion and transmission, Image handling etc.). We also discussed the application development environment and implementation issues.

In Chapter 5, we will run experiments and study techniques on improving the performance of the application

CHAPTER 5

EXPERIMENTS AND RESULTS

Based on discussions in Chapter 3 and based on the features of the mobile map application, we identify some promising techniques (prefetching and caching) to integrate into the mobile map application. We study the effects of the chosen techniques to the mobile map application. The techniques are chosen aimed at better application performance, that is shorter user perceived latency and efficient wireless bandwidth usage.

5.1 Problem Analysis and Study Objective

Due to low wireless bandwidth and slow palm processing ability, a user may suffer long latency. Our study objective is to find techniques to improve the application performance and at the same time efficiently use the precious wireless bandwidth.

User perceived latency is the time interval from the time a user is sending a request until the time the required map is displayed on the palm screen. First we analyze the factors of user perceived latency. In the mobile map application, big maps are stored in the server as Image objects. Each time when the client requests a new map segment, the server filters out the required map segment from its big maps, encodes the map segment into GIF data, and sends it to the client. At the client side, the received GIF data is converted into an Image object and displayed on the palm screen. The user perceived latency is mainly composed of three factors: map data transmission time, map data processing time and map displaying time. Map data transmission time is the time interval to transmit the map GIF data of a map segment from server to client. Map data processing time is the time interval to decompress map GIF data of a map segment and convert it into an Image object. Map displaying time is the time interval to display a map segment on the palm computer screen.

The following figure shows the end-to-end processing of a map segment from server to client.

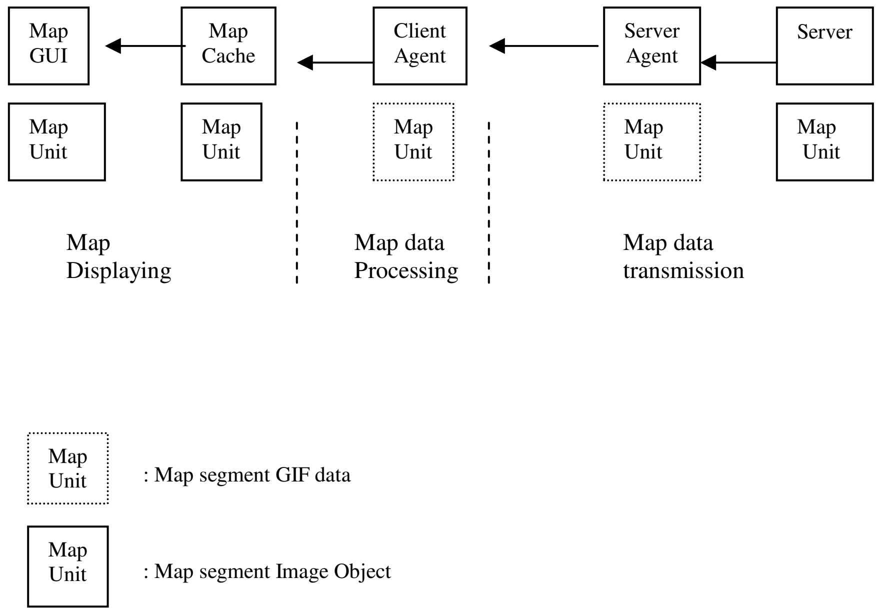


Figure 5.1. Map Unit End-to-end Processing

The map displaying time is very short and can be neglected. The transmission bandwidth between client and server is low which causes long map transmission times, and the process for the client to decompress and convert GIF data into an image object takes a long time. Therefore users experience long latency to see the required map segment. Finding good techniques to reduce user perceived latency and at the same time, efficiently use wireless bandwidth is the objective of our study in this chapter.

5.2 Solutions – Prefetching and Caching

To reduce user perceived latency, caching and prefetching techniques are used, and when to execute map data processing is carefully considered. From the discussion of Chapter 3, the basic principle of choosing a prefetching and caching schemes is that: prefetch data which is most likely to be used soon, whenever necessary, replace the data in cache which is the most unlikely to be used in the future and make room for the new arriving data.

Our mobile map is a position sensitive mobile application. Given the current user location and the current map segment, the map segments adjacent to the current map segment are the ones most likely to be required next, and will be prefetched. We study two prefetching schemes: directional and non-directional. The map segment in the cache that is the farthest to the current map segment is the one that is most unlikely to be used in the near future, and will be replaced when necessary.

The prefetching and caching schemes are closely related to the map cache architecture. First we describe the map cache architecture and define the distance between two map segments.

Map segments in the cache are organized into 5 vectors corresponding to the five map resolution levels. For example, map segments of level 0 are stored in vector0. For each vector, map segments are added by incoming order. The following figure shows the architecture of the map cache.

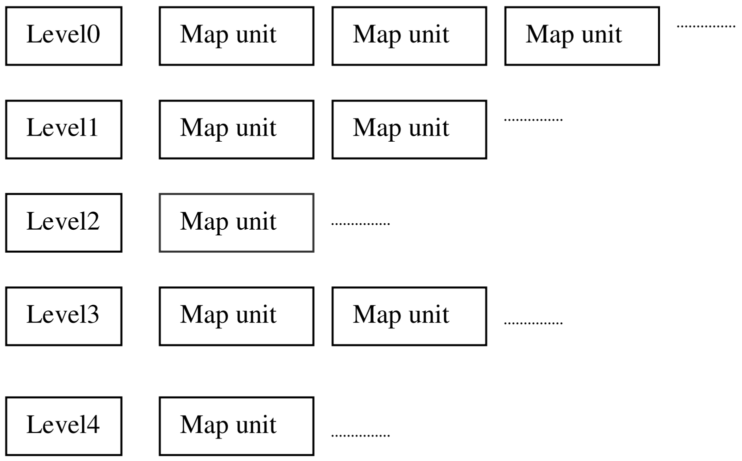


Figure 5.2: Map Cache Architecture

Each map segment is of fixed size (60*50 pixels, more in Section 5.4). Map segments of different resolution levels represent different size real areas. We call the area represented by a map segment with level n an area segment of level n. Map segments of higher resolution level represent smaller area segments, map segments of lower resolution level represent bigger area segments.

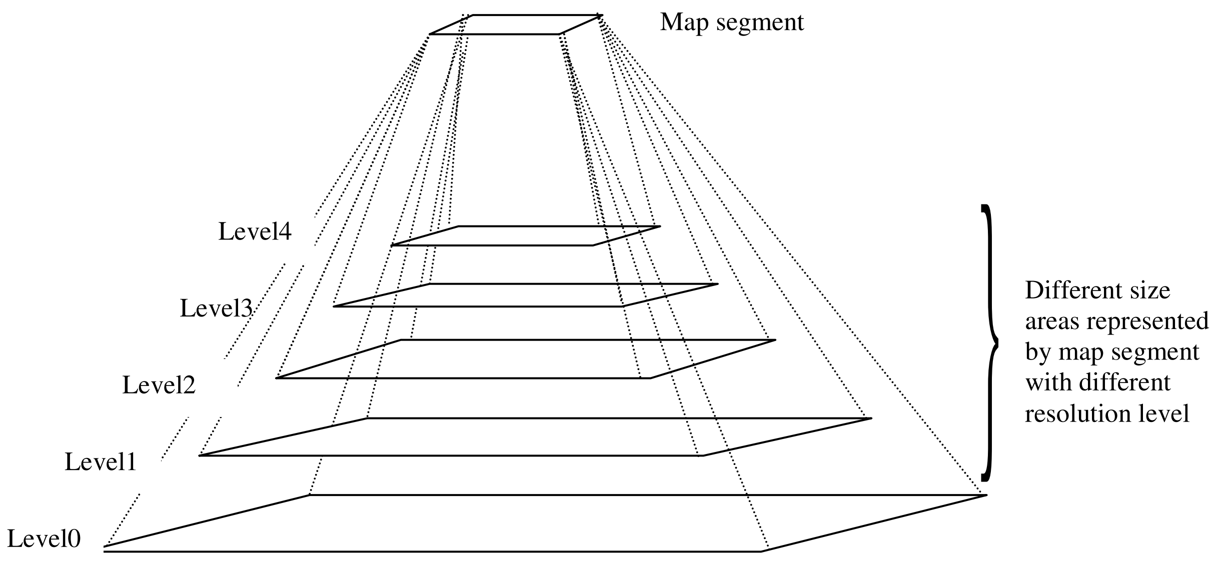

Map segment
Level4
Level3
Level2
Level1
Level0
Different size areas represented by map segment with different resolution level

Figure 5.3. Map Segments of Different Resolution Levels

Represent Different Size Real Areas

The distance between the current map segment and the other map segments in the cache plays a key role in our prefetching and caching schemes. We define this distance as follows:

Assume A(x0, y0, lev0) represent the current map segment (coordinates of the center location and resolution level), B(x, y, lev) represent other map segments in the cache. Let d(A, B) be the distance between the two map segments.

Assume the latitude increment of a level n area segment is incXn, the longitude increment of a level n area segment is incYn. We define the horizontal distance h(A, B) between the two map segments at level lev0 to be the number of map tiers around A(x0, y0, lev0) until the map segment centered at (x, y).

h(A, B) = max (round((x-x0)/incXlev0), round((y-y0)/incYlev0))

For example, in the following figure, h(A, B) is 3. We define the vertical distance v(A, B) to be the absolute value of the difference between lev0 and lev.

v(A, B) = abs(lev0-lev)

Finally we define: d(A, B) = h(A, B) + v(A, B)

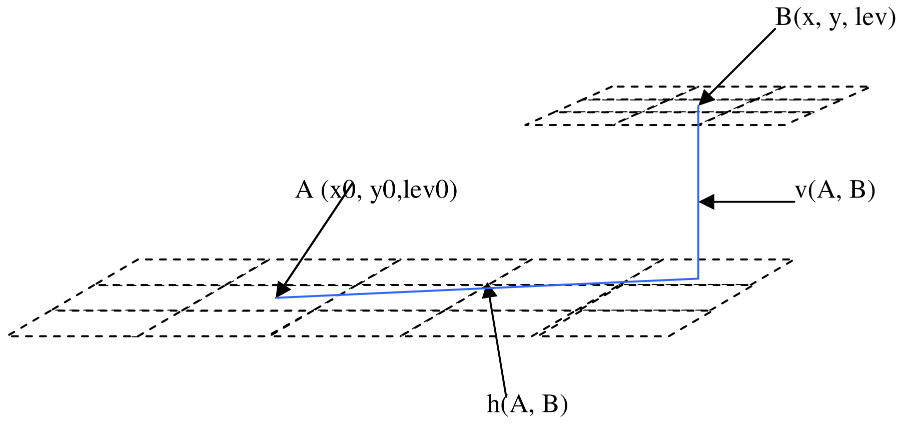


Figure 5.4. Distance between the current map segment and other map segments in Cache

Next we describe the details of our prefetching and caching schemes.

5.2.1. Caching

Within the memory limit, cache as much as possible, replace the map segment, which is farthest to the current user location when necessary.

At the client side, map segments received or prefetched from the server are stored in the map cache. Based on the memory space of the palm computer and map segments size, we set the maximum map cache size to be 50 map units (more in Section 5.4). Within this limit, we cache as much as possible. When the map cache is full and there is a new map segment, we choose to remove the map segment in the cache that is farthest to the current location and to make room for the new map segment.

Cache Replacement Algorithm:

```
//Find the map segment in the cache that is farthest to the current map segment and remove it.
Current map unit:  A(x0, y0, lev0) (Coordinates of center location and resolution level)
int maxdist=0;
int distance=0;
int maxi=0, maxj=0. // (vector index and position index of the farthest map segment in the cache
for (int i=0, i<5, i++)
        {
        for (int j=0; j<level(i).length; j++)
                {
```

```
                MapUnit B(x, y, lev)=vector(i)(j)
                dist=d(A, B)  //as defined before
                if dist >maxdist
                {
                        maxdist=dist
                        maxi=i;
                        maxj=j;
                }
        }
}
remove element at position (maxj) from vector(maxi)
```

5.2.2 Prefetching - Non-Directional

Prefetch map segments that are adjacent to the current map unit.

Based on the current user location and map resolution level, we prefetch map segments which are adjacent horizontally (same level) or vertically (one level above or below) to the current map segment. In this scheme, we neglect user direction. This scheme should be suitable for the scenarios where users roam around randomly.

Assume the current map segment is A(x0, y0, lev0). At level lev0, we prefetch four map segments which are adjacent to A(x0, y0, lev0). At level lev0-1, we prefetch one map segment which is centered at (x0, y0). At level lev0+1, we prefetch the map segment centered at (x0, y0) and 4 map segments adjacent to it.

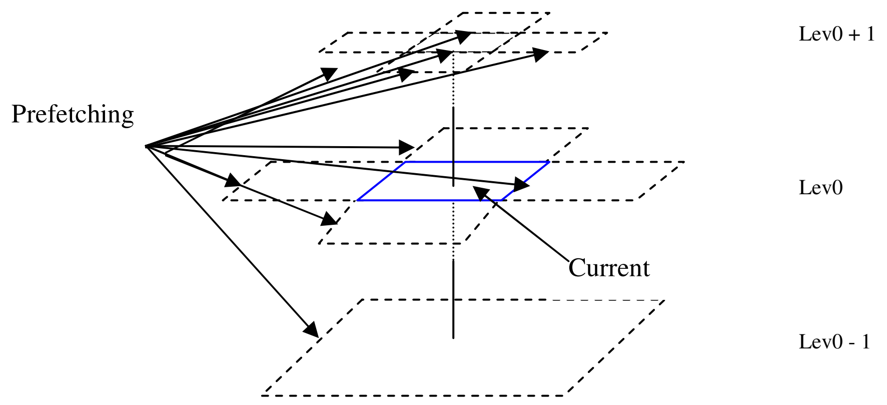


Figure 5.5. Non-directional Prefetching

Prefetching Algorithm:

```
Current mapunit : A(x0, y0, L0)
Prefetch (x0+incX0, y0+incY0, L0)
Prefetch (x0-incX0, y0+incY0, L0)
Prefetch (x0+incX0, y0-incY0, L0)
Prefetch (x0-incX0, y0-incY0, L0)
If (L0-1>=0)
        Prefetch (x0, y0, L0-1)
If (L0+1<=4)
{
        Prefetch (x0, y0, L0+1)
        Prefetch (x0+incX0, y0+incY0, L0+1)
        Prefetch (x0-incX0, y0+incY0, L0+1)
        Prefetch (x0+incX0, y0-incY0, L0+1)
        Prefetch (x0-incX0, y0-incY0, L0+1)
}
```

5.2.3 Prefetching Scheme 2- Directional

Prefetch map segments adjacent to the current map segment in the current user direction.

In this prefetching scheme, we take user direction into account. Based on the current user location and map resolution level, we prefetch map segments which are adjacent horizontally (same level) or vertically (one level above or below) to the current map segment along the current user direction. This scheme should be suitable for scenarios where users move along a certain direction.

Assume the current map segment is A(x0, y0, lev0, dir0): coordinates of center location, current resolution level and direction. The value of dir0 could be NE (northeast), NW (northwest), SW (southwest), or SE (southeast).

If dir0=NE, at level lev0, we prefetch three map segments which are adjacent to A(x0, y0, lev0) in the NE direction. At level lev0-1, we prefetch one map segment which is centered at (x0, y0). At level lev0+1, we prefetch the map segment centered at (x0, y0) and 3 map segments adjacent to it toward direction dir0. The following figure shows the prefetching scheme when dir0=NE.

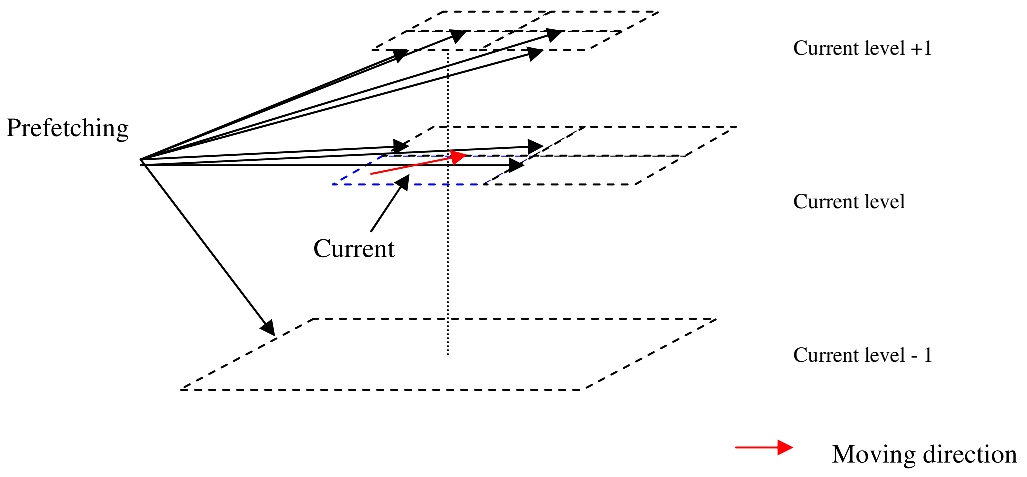


Figure 5.6. Directional Prefetching Scheme

(Current Direction=NE)

For the cases where dir0 takes other values, the prefetching scheme is the same as above, prefetching only map segments along the appropriate direction.

Prefetching Algorithm:

```
Current mapunit A(x0, y0, L0) (coordinates of center point and level)
If direction=NE
        Prefetch (x0-incX0, y0, L0)
        Prefetch (x0-incX0, y0+incY0, L0)
        Prefetch (x0, y0+incY0, L0)
        If (L0-1>=0)
                Prefetch (x0, y0, L0-1)
        If (L0+1<=4)
        {
```

```
        Prefetch (x0, y0, L0+1)
        Prefetch (x0-incX0, y0, L0+1)
        Prefetch (x0-incX0, y0+incY0, L0+1)
        Prefetch (x0, y0+incY0, L0+1)
    }
If direction=NW
    Prefetch (x0+incX0, y0, L0)
    Prefetch (x0+incX0, y0+incY0, L0)
    Prefetch (x0, y0+incY0, L0)
    If (L0-1>=0)
        Prefetch (x0, y0, L0-1)
    If (L0+1<=4)
    {
        Prefetch (x0, y0, L0+1)
        Prefetch (x0+incX0, y0, L0+1)
        Prefetch (x0+incX0, y0+incY0, L0+1)
        Prefetch (x0, y0+incY0, L0+1)
    }
If direction=SW
    Prefetch (x0-incX0, y0, L0)
    Prefetch (x0-incX0, y0-incY0, L0)
    Prefetch (x0, y0-incY0, L0)
    If (L0-1>=0)
```

```
                Prefetch (x0, y0, L0-1)
        If (L0+1<=4)
        {
                Prefetch (x0, y0, L0+1)
                Prefetch (x0-incX0, y0, L0+1)
                Prefetch (x0-incX0, y0+incY0, L0+1)
                Prefetch (x0, y0+incY0, L0+1)
        }
If direction=SE
        Prefetch (x0+incX0, y0, L0)
        Prefetch (x0+incX0, y0-incY0, L0)
        Prefetch (x0, y0-incY0, L0)
        If (L0-1>=0)
                Prefetch (x0, y0, L0-1)
        If (L0+1<=4)
        {
                Prefetch (x0, y0, L0+1)
                Prefetch (x0+incX0, y0, L0+1)
                Prefetch (x0+incX0, y0-incY0, L0+1)
                Prefetch (x0, y0-incY0, L0+1)
        }
```

5.2.4. Timing of Map Data Processing

Map data processing time is a prominent factor in user perceived latency. During the process from client receiving the GIF data of a map segment until the map is displayed on the screen, when to execute the map data processing (decompress and convert GIF data to an Image object) has a big effect on the user perceived latency. We considered two options. The first is to put the process just after the data is received by the client agent, and store the resulting Image object in the map cache. The second option is to put the process before displaying the map, cached maps are in GIF data format. For the second option, every map needs to be processed before it can be displayed. The user perceived latency is at least as long as the map data processing time, which is a long latency. For the first option, the maps in the cache are Image objects and ready to be displayed. When a user request is met in the cache, the required map segment is displayed immediately, and the user perceived latency is just the map displaying time and is very short.

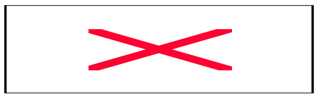
On the other hand, when to process the received GIF map data in the client also affects the usage of the memory space. Since the GIF data of a map segment is in compressed form, it takes smaller space than a map segment as an Image object. Hence the option two which stores GIF data in cache gives better memory utilization than the option one which stores Image objects in the cache.

So, each option has advantages and disadvantages. We have to choose one. In the mobile map application, where techniques to reduce user perceived latency is our main objective, we choose option one.

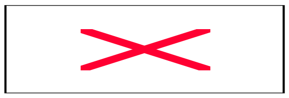
5.3 Mobile Map Performance Criteria

To study the effects of the chosen techniques on the mobile map application, we must first decide how to measure the effect of each technique and the performance of the application. We consider the following criteria: *user perceived latency, request meet rate, cache hit rate, and wireless bandwidth efficiency.*

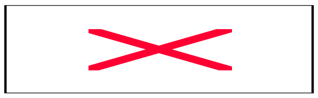
- User perceived latency (UPL): time interval between the client sending a request until receiving the required map – measures how fast a user's request is met, the shorter the better.
- Request meet rate (RMR): ratio of the number of maps displayed to the number of total requests – measures how many user requests are met, the bigger the better.



- Cache-hit-rate (CHR): ratio of the number of requests met in cache to the number of total requests – measures how efficient the caching and prefetching algorithms are, the bigger, the better.



- Bandwidth efficiency (BE): ratio of the number of map units used to the total number of map units fetched from server - measures how efficiently the wireless bandwidth is used, the bigger the better.



5.4 Experiment Environment and Assumptions

The server runs on Windows 98. The client runs on a palm emulator. The tracing of user locations by GPS is emulated by using a user trace log file that contains a series of GPS readings. The data from the trace log file is pre-stored in the palm emulator database. When the application is running, the client continuously reads in the user locations from the database, requests map segments from the server, and displays the user trace and map segment on the palm emulator screen. The user activities of changing map resolution level is emulated by inserting statements to change map resolution levels in the execution it a random pattern. By doing this, the same trace file and resolution level changing pattern can be repeatedly used to test different caching and prefetching schemes, and reasonable comparisons of the results can be made.

5.4.1. Client memory space, map segment size and map cache size

Client program memory space is 220 kbytes. The program takes about 50 kbytes. Within this restriction, how to set the map segment size has a fundamental effect on the experiments. Initially, we set the map segment size to be the full palm screen size, that is 160*150 pixels (excluding the buttons panel). But the palm memory can hold at most two map segments of full color and of size 160*150 pixels, the user perceived latency of such a map segments is about 5 minutes and most of the time the user will not see the required map because he has moved out of the current map boundary before the map segment is retrieved. And hence there is no way to test any prefetching and caching schemes.

Consider our objective to study the effects of prefetching and caching schemes, a certain amount of map segments has to be accommodated in the cache. Through experiments, we choose the map segment size to be black-and-white 60*50 pixels (width * height), the map cache size to be 50 map segments.

We keep the idea in mind that, in the future when the palm technology is improved to allow more full palm screen size map segments to be stored in the memory, the application and all the experiments still work with little modifications.

5.4.2 Transmission Bandwidth between Client and Server

We estimate the transmission bandwidth between client and server by transmitting a certain amount of data from the server to the client and recording the transmission time. Since we are using the palm emulator, we noticed that the emulator recorded time is different from real time. The following table displays the experiment data.

Table 5-1. Transmission Bandwidth Estimation

Experi-ment #	Start time	End time	Time interval	Data amount	Computer time
1	12:06:18	12:15:05	8.47(min)	600,000 (bytes)	3059720 (milliseconds) = 51 minutes
2	12:27:20	12:31:45	4.25 (min)	300,000(bytes)	1540380(milliseconds) = 25.7 (minutes)
3	12:39:35	12:40:03	0.28 (min)	30,000(bytes)	175150 (milliseconds) = 2.9 (minutes)

In the above table, the left three columns are hand recorded time, the last column is emulator recorded time. Estimated emulator recorded time to real time ratio is 6:1.

In the following we calculated the estimated bandwidth.

Bandwidth=(data amound transmitted) / (transmission time)

To eliminate the initial setup effect, we omit the transmission of the first 30,000 bytes of data.

In experiment 1:

Data amount transmitted = 600,000-30,000=570,000 (bytes)=570,000*8=4,560,000 (bits)

Transmission time = 51-2.9=48.1 (min, emulator time) =48.1/6=8 (min, real time)=8*60=480 (sec, real time)

Bandwidth = 4,560,000/480=9500 bps

In experiment 2:

Data amount transmitted = 300,000-30,000=270,000 (bytes)=270,000*8=2,160,000(bits)

Transmission time=25.7-2.9=22.8 (min, emulator time)=22.8/6=3.8 (min, real time)= 3.8*60=228 (sec, real time)

Bandwidth = 2,160,000/228=9474 bps

From the above calculation, the estimated bandwidth is about 9500bps, which reasonably emulates the current real world wireless bandwidth in 2G wireless systems.

5.5 Experiments and Results

In these section, we design and run experiments to study the effect of the chosen caching and prefetching schemes described in Section 5.2 to the mobile map application. We run the following five experiments, each will be repeated using 3 different user trace log files named log1(size: 102 kbytes), log2 (size: 268 kbytes), and log3 (size: 309 kbytes).

- Experiment1 (Exp1): No caching and no prefetching
- Experiment2 (Exp2): only caching
- Experiment3 (Exp3): only directional prefetching
- Experiment4 (Exp4): Caching and non-directional prefetching
- Experiment5 (Exp5): Caching and directional prefetching.

For each experiment, we record the following data:

- User perceived latencies
- Number of total requests

- Number of total maps transmitted
- Number of cache hits
- Number of maps displayed

5.5.1 Experiments

In this section, each experiment and its purpose are briefly explained.

Experiment 1. No Caching and no Prefetching

In this experiment, we run the simplest version of the mobile map application. There are no prefetching and caching schemes integrated. When a user requires a map, the map segment will be retrieved from the server. User perceived latency includes map transmission time and map processing time. Cache hit rate is zero. The results will be used as a base comparison reference to study other techniques.

Experiment 2. Only Caching

In this experiment, the caching technique is used in the application. Whenever a map segment is retrieved from the server, it is stored in the client map cache. When a user requires a map segment, it is searched in the cache first, if found, use it, otherwise retrieve it from the server. When the cache is full, the map replacement algorithm described in Section 5.2 is used to make room for new map segments. This experiment is to study how much improvement the caching only scheme brings to the application.

Experiment 3. Only Directional Prefetching

In this experiment, we implement the directional prefetching scheme in the application. Map segments fetched and prefetched from server are stored in the cache. When a map segment is used/displayed, it is removed from the cache. The purpose of this experiment is to test the effect of the prefetching-only scheme on the application.

Experiment 4. Caching and Non-directional Prefetching

In this experiment, both caching and non-directional prefetching schemes are integrated in the application. The purpose is to test the combination effect of the two techniques.

Experiment 5. Caching and Directional Prefetching

In this experiment, caching and directional prefetching schemes are integrated in the application. The purpose is to test the combination effect of the two techniques.

5.5.2. Results

The results of the five experiments on 3 log files, totally 15 experiments, are collected and displayed in the following tables and figures.

The following table displays the various counts from each experiment.

Table 5-2. Experimental Results (1)

		Log1	Log2	Log3
Exp1	Total requests	58	70	87
	Maps transmitted	28	53	62
	Maps used	8	24	35
	Cache hits	0	0	0
Exp2	Total requests	58	70	87
	Maps transmitted	13	17	16
	Maps used	36	48	67
	Cache hits	33	47	63
Exp3	Total requests	58	70	87
	Maps transmitted	37	95	119
	Maps used	22	41	57
	Cache hits	14	29	38
Exp4	Total requests	58	70	87
	Maps transmitted	36	70	100
	Maps used	43	55	74
	Cache hits	39	58	72
Exp5	Total requests	58	70	87
	Maps transmitted	34	71	94
	Maps used	43	54	75
	Cache hits	42	54	74

The following table contains the calculated performance criteria based on Table 5-2 for each experiment.

Table 5-3. Experimental Results (2)

(All results are in percentage, %)

	Log1			Log2			Log3		
	RMR	CHR	BE	RMR	CHR	BE	RMR	CHR	BE
Exp1	13.79	0	28.57	34.29	0	45.28	40.23	0	56.45
Exp2	62.07	56.9	276.92	68.57	67.14	282.35	77.01	72.41	418.75
Exp3	37.93	24.14	59.46	58.57	41.43	43.16	65.52	43.67	47.90
Exp4	74.14	67.24	119.44	78.57	82.86	78.57	85.06	82.76	74.00
Exp5	74.14	72.41	126.47	77.14	77.14	76.06	86.21	85.06	79.79

The following table shows the average UPL for each experiment.

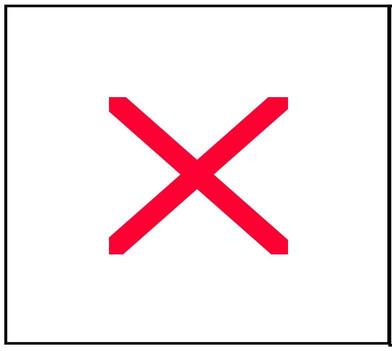
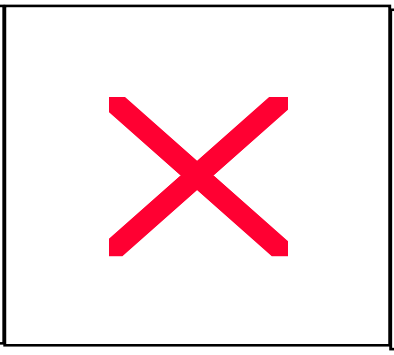
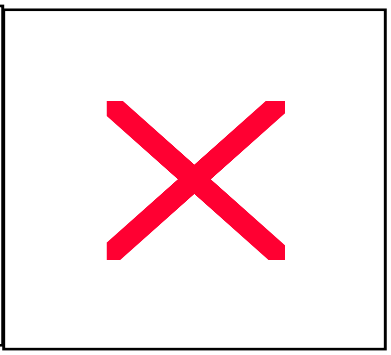
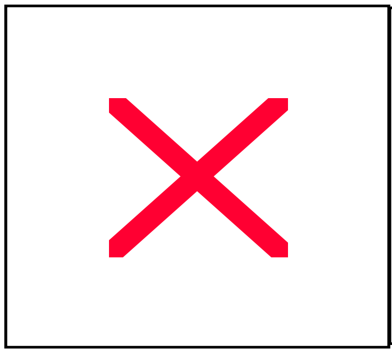
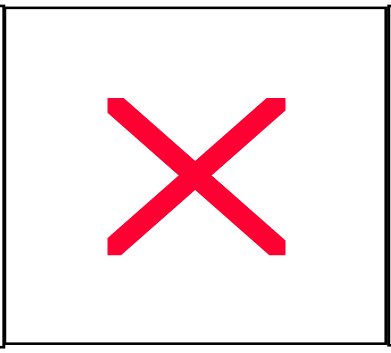
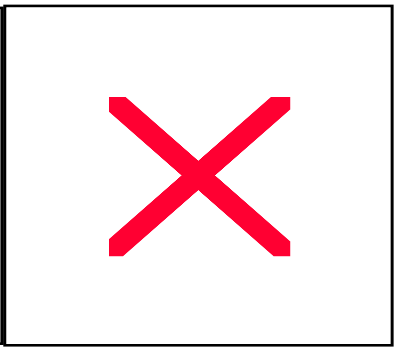
Table 5-4. Average UPLs

(Time unit: millisecond, Emulator time)

	Log1	Log2	Log3	Average (sec, real time)
Exp1	175760	171973	167248	28.61
Exp2	6823	12105.43	8477	1.52
Exp3	29541	46262	57906	7.42
Exp4	6265	3661	4503	0.80
Exp5	3784	10861	3695	1.02

(Note: when calculating the average UPL, the first two UPLs are neglected to avoid the initial setup effect)

The following figure shows the requests percentage distribution with regard to UPL.

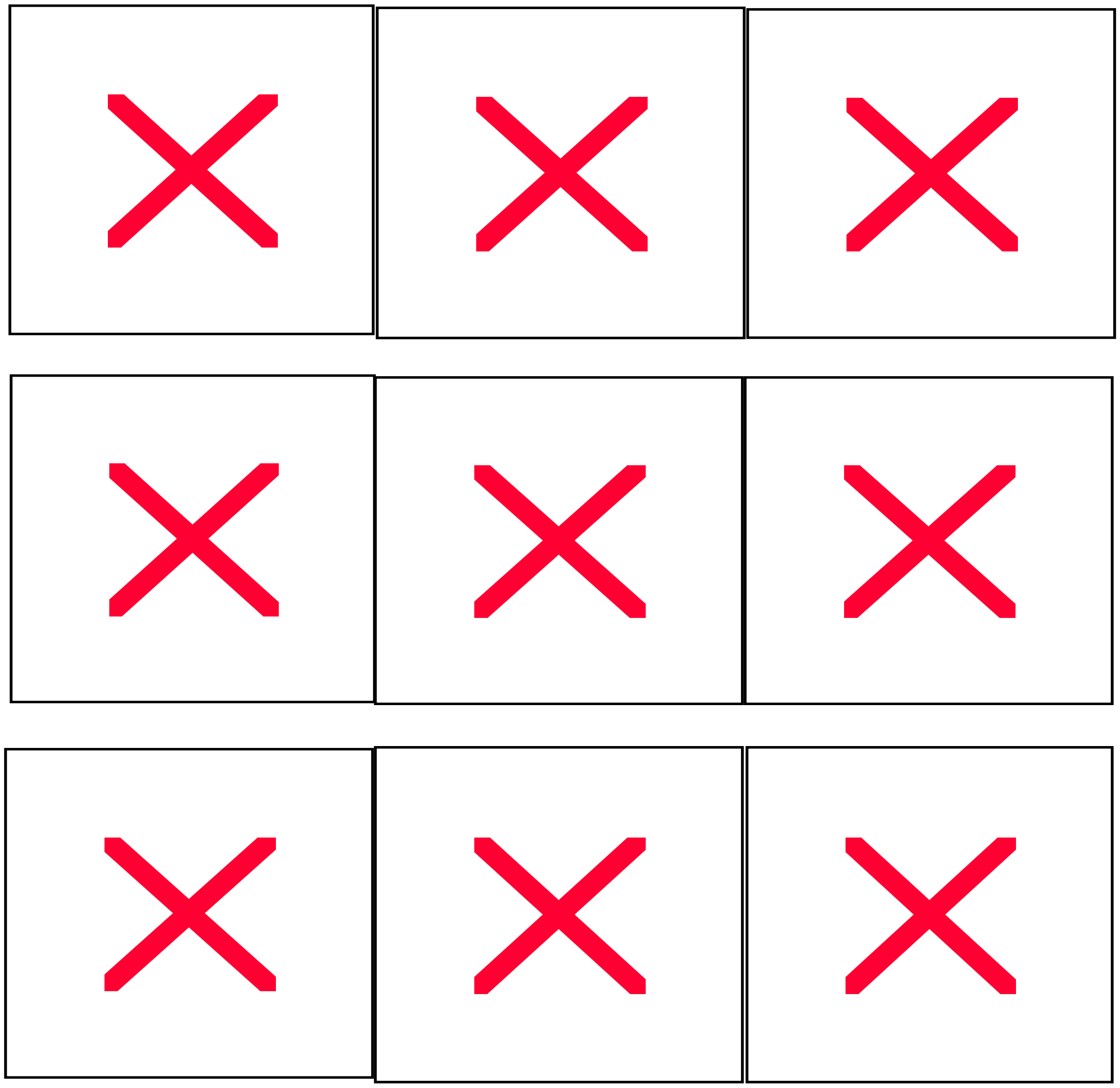


Figure 5.7. Requests Distribution with regard to UPL

Figure 5.7 explanation:

- Missed: the percentage of requests that are missed, i.e. request is not sent to server and no map segment is retrieved.

- 0_5000: the percentage of requests which are met with UPL in the time range 0-5000 milliseconds (emulator time). These requests are met by the cache.
- 5000-100000: the percentage of requests which are met with UPL in the time range 5000-100000 milliseconds (emulator time). These are the cases where when the request is initiated, the required map segment is already being prefetched but only half way in the transmission or processing.
- >100000: the percentage of requests which are met with UPL in the time range >100000 milliseconds. These are the cases where the UPL covers the whole map transmission and processing time.

5.5.3. Results Analysis

We analyze the experiment results from three aspects. From the user's perspective, Exp1 has an RMR of about 30% and an average UPL of about 28 seconds. Exp2 has an RMR of about 70% and an average UPL of 1.52 seconds. Exp4 and Exp5 have very similar results. They have an RMR of about 80% and an average UPL about 1 second. Exp3 has an RMR of about 55% and an average UPL of 7.42 seconds. We see that Exp2, Exp4 and Exp5 provide outstanding improvement in user satisfaction. Exp3 provides a moderate improvement in user satisfaction.

From the service provider's perspective, Exp1 gives a BE in the range of 30%-50%, Exp2 gives a BE in the range of 300%-400%, Exp4 and Exp5 give a BE in the range of 80%-120%. Exp3 gives a BE about 50%. Obviously, Exp2 provides the best result.

Considering the efficiency of caching and prefetching schemes, Exp2 gives a CHR of about 65%, Exp4 and Exp5 give a CHR of about 75%, Exp3 gives a CHR of about 35%.

Figure 5.7 gives us a clear indication of how many requests are missed, how many requests are hit in the cache, and how many requests are met from the server in each of the experiments. It shows that Exp2, Exp4 and Exp5 provide similar and better user satisfaction.

Concluding from the above analysis, Experiment 2, the caching only approach, gives a better solution than other approaches overall. The reason behind this is that the user trace files we are using are in a roaming around pattern within a certain area.

It is obvious that the features of the user trace have big impact on the experiment results. If the user roams around without a clear destination (like in a tourist site), caching and non-directional prefetching could bring better result. If the user moves toward a certain destination, directional prefetching could give a good result. It is expected that the experimental results will be closely related to the trace file pattern and features.

5.6 Summary

In this chapter, we chose one cache replacement scheme and two prefetching schemes and integrated them in the mobile map application. We designed 5 experiments and repeated each on three user trace files to test the effects of these techniques on the application performance. It was found that the caching only approach gives a better solution in both user satisfaction and wireless bandwidth utilization. This is based on the fact that the user trace file we are using is in a roaming around pattern within a certain area. It is expected that the experimental results will be closely related to the trace file pattern and features.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This chapter concludes this thesis and provides recommendation for future work

6.1 Contributions

The work carried out in this thesis can be described as a pyramid. The pinnacle is the final goal and core task of the thesis, and contains a specific mobile computing application – a mobile map system, particularly identified techniques for the application, and simulations and results analysis to study the impact of the chosen techniques on the performance of the application. The bottom layer is the foundation of the thesis. It provides a background introduction to mobile computing, including architecture and components of mobile computing systems, wireless network technologies and services, mobile computers, and mobile software systems. It covers knowledge ranging from hardware to software, from technologies to available products. The middle layer is dedicated to a review of existing techniques in improving performance of mobile computing applications. These techniques include caching, prefetching and compression. This layer sits on the bottom foundation and provides a collection of techniques from which the top layer identifies the promising techniques to apply to the mobile map application.

The contributions of this thesis are summarized as follows:

1. A literature review on existing techniques and researches in improving performance of mobile computing applications has been carried out. We concentrated on techniques of caching, prefetching and compression. These are the most commonly used techniques to improve transmission latency and wireless bandwidth utilization.
2. A new client/server mobile computing application - a Mobile Map System - has been developed. The mobile map system traces a user's move with assistance of the GPS system, automatically downloads maps from an Internet map server to handheld computers via wireless networks, and displays user trace and map on the handheld computer screen. This application is developed using OOD, OOP and Java KVM technologies.
3. Based on the techniques studied in the literature review and the features of the mobile map application, we identified some promising techniques and integrated them into the mobile map application. Through experiments, we studied the effects of these techniques to the application performance. We concluded that the caching only approach provides a better solution than others. This is based on the trace files we are using.

6.2 Future Work

The following areas are recommended for future study.

- Extending the mobile map application and performance study to different user groups. Mobile map users can be categorized by fast moving, slow moving, directional moving, roaming around, highway long distance driving, small area wandering. The current mobile map application is executed based on user trace files with a moderate speed and within a limited area. It is obvious that the mobile map application and integrated techniques will perform differently for different user groups. For example, for highway drivers, low resolution maps covering big areas are most likely needed, directional prefetching should be a suitable technique for this case. For users at a tourist site, high resolution maps are needed, caching is a good technique. How to make this application adaptive to different user groups and perform consistently needs further study and work.
- Studying incremental map download methods and applying this approach to the mobile map application, so that the application performs more user-friendly. For each map request, a high-level map segment will be transferred and displayed first with more map data been incrementally transferred and added to the current map. This will increase the RMR (which is still fairly low, see Table 5-3) and hence increase the user's satisfaction.

- Studying new techniques to improve performance of mobile computing applications. Due to the limitation of time and space of this thesis, it is impossible to review every technique. There could be other existing techniques not covered in this thesis or new techniques that can be applied to the mobile map applications to improve its performance.
- Studying the opportunities, problems and issues to transfer the mobile map application into a commercial product. We wish that in the future the mobile map become a common application in mobile computers so that people on-the-road will never worry to get lost.

Mobile computing is an area that is still young and with a huge potential future market. We expect that in the future handheld computers become a necessary possession in people's pocket. With abundant applications available for mobile computers and with the rich and high quality services of mobile service providers, people will realize the dream of connecting anywhere and anytime at will.

REFERENCES

[AHSON98] S. A. Ahson, I. Mahgoub, "Research Issues in Mobile Computing", Proceedings of 1998 IEEE International Performance, Computing, and Communication Conference, Phoenix/Tempe, Arizona, Feb. 16-18, 1998, p. 209 –215.

[BAEN97] M. Baentsch, L. Gaum, G. Molter, S. Rothkugel, P. Sturm, "Enhancing the Web's Infrastructure: From Caching to Replication", IEEE Internet Computing, Vol. 1, No. 2, Mar. – Apr. 1997, p.18-27.

[BARB99] Daniel Barbara, "Mobile Computing and Databases – A Survey", IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 1, Jan. – Feb. 1999, p.108-117.

[BASS90] M.A. Bassiouni, A. Mukherjee, "Data Compression in Real-Time Distributed Systems", IEEE GLOBECOM '90, San Diego, California, Vol. 2, Dec. 1990, p.967-971.

[BELL98] A. Belloum, L. O. Hertzberger, "Document Replacement Policies Dedicated to Web Caching", Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference, Gaithersburg, Maryland, Sept. 14-17, 1998, p.576-581.

[CAO98] P. Cao, C. Liu, "Maintaining Strong Cache Consistency in the World Wide Web", IEEE Transactions on Computers, Vol. 47, No. 4, Apr. 1998, p.445-457.

[CAO97] P. Cao, S. Irani, "Cost-aware WWW Proxy Caching Algorithms", Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems, Monterey, California, Dec. 1997, p. 193-206.

[CHAN99] M. C. Chan, T. Y. C. Woo, "Next-Generation Wireless Data Services: Architecture and Experience", IEEE Personal Communications, Feb. 1999, p. 20-33.

[CHIO00] Hann-Huei Chiou, A. I-Chi Lai, Chin-Laung Lei, "Prediction-Capable Data Compression Algorithms for Improving Transmission Efficiency on Distributed Systems", Proceedings of the 20th International Conference on Distributed Computing Systems, Taipei, Taiwan, Apr. 10–13, 2000, p. 654-661.

[COHN99] E. Cohen, B. Krishnamurthy, J. Rexford, "Efficient Algorithms for Predicting Requests to Web Servers", Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, New York, New York, Mar. 21-25, 1999, p. 284 –293.

[FASB99] A. Fasbender, F. Reichert, " Any Network, Nay Terminal, Anywhere ", IEEE Personal Communications, Apr. 1999, p.22-30.

[FORM94] G. H. Forman, J. Zahorjan, "The Challenges of Mobile Computing", IEEE Computer, Vol. 27, No.4, Apr. 1994, p.38-47.

[HILL96] A. Hills, D. B. Johnson, "Wireless Network Infrastructure at Carnegie Mellon University", IEEE Personal Communications, Vol. 3, No. 1, Feb. 1996. p. 56-63.

[HORI] "Horizontal Application Index", http://www.mobileinfo.com/Applications_Horizontal/index.htm

[IBRA00] T. I. Ibrahim, Cgeng-Zhong Xu, " Nueral Nets Predictive Prefetching to Tolerate WWW Latency", Proceedings of the 20th International Conference on Distributed Computing Systems, Taipei, Taiwan, Apr. 10 – 13, p. 636 –643.

[IMIE96] T. Imielinski, H. F. Korth, “Mobile Computing”, The KLUWER International Series in Engineering and Computer Science, Vol. 353, Kluwer Academic Publishers, ISBN: 0792396979, Boston, 1996.

[INTR] “Introduction to Wireless LAN”, http://www.wlana.com/intro/index.html

[IRAN97] S. Irani, “Page Replacement with Multi-size Pages and Applications to Web Caching”, Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, El Paso, Texas, May 4-6, 1997. p. 701-710,

[JACK93] David Jeff Jackson, Sidney Joel Hannah, “Comparative Analysis of Image Compression Techniques”, Proceedings of the Twenty-fifth Southeastern Symposium on System Theory, Tuscaloosa, Alabama, Mar. 1993, p. 513 –517.

[JIANG98a] Z. Jiang, L. Kleinrock, “Web Prefetching in a Mobile Environment”, IEEE Personal Communications, Oct. 1998, p. 25-33.

[JIANG98b] Z. Jiang, L. Kleinrock, “An Adaptive Network Prefetch Scheme”, IEEE Journal on Selected Areas in Communications, Vol. 16, No. 3, Apr. 1998, p 358-368.

[LAYN99] P. Layne, “Software Architecture For Mobile Computing”, Masters Thesis, Dept. of Computer Science and Information Systems, Univ. of Jyvaskyla, Finland, Oct. 1999.

[LETT99] P. Lettieri, M. B. Srivastava, “Advances in Wireless Terminals”, IEEE Personal Communications, Feb.1999, p.6-19.

[NEIL93] E. J. O’Neil, P. E. O’Neil, G. Weikum, “The LRU-K Page Replacement Algorithm for Database Disk Buffering”, Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, May 26-28, 1993, p. 297-306.

[KUNZ99] T. Kunz, J. P. Black, "An Architecture for Adaptive Mobile Applications", Proceedings of Wireless 99, the 11th International Conference on Wireless Communications, Calgary, Alberta, Canada, July 1999, p. 27-38.

[RACH96] G. Racherla, A. Das, "Mobile Computing, A Promising Future That Still Requires Much Work", IEEE Potentials, Vol. 15, No. 4, Oct.- Nov. 1996, p.13 –15.

[REDD98] M. Reddy, G. P. Fletcher, "An Adaptive Mechanism for Web Browser Cache Management", IEEE Internet Computing, Vol. 2, No. 1, Jan.- Feb. 1998, p.78 –81.

[RYSA99a] P. Rysavy, "General Packet Radio Services (GPRS)", http://www.MDI-ng.org/es53061/3G.htm

[RYSA99b] P. Rysavy, "The evolution of Cellular Data", http://www.MDI-ng.org/documentation.html

[SATY]: M. Satyanarayanan, "Mobile Computing", IEEE Computer, Vol. 26, No. 9, Sept. 1993, p.81-82.

[SCHE97] P. Scheuermann, J. Shim, R. Vingralek, "A Case for Delay-Conscious Caching of Web Documents", Computer Networks and ISDN Systems, Vol. 29, No. 8-13, Sept. 30, 1997, p. 997-1005.

[SHIM99] J. Shim, P. Scheuermann, R. Vingralek, "Proxy Cache Algorithms: Design, Implementation, and Performance", IEEE Transactions on Knowledge and Data Engineering, Vol. 1, No. 4, July-Aug. 1999, p.549-562.

[WANG96] Zheng Wang, J. Crowcroft, "Prefetching in World Wide Web", Proceedings of the IEEE Global Internet 96, London, UK. Nov.1996, p. 28-32.

[WEIS91] M. Weiser, " The Computer for the 21st Century", Scientific American, Vol. 265, No. 3, Sept. 1991, p. 94-104.

[WILL97] S. Williams, M. Abrams, C. Standridge, G. Abdulla, E. Fox, “Removal Policies in Network Caches for World-Wide-Web Documents’, Proceedings of the ACMSIGCOMM, Palo Alto, California, Aug. 1997, p. 293-305.

[YOUN91] N. Young, “On-line Caching as Cache Size Varies”, Proceedings of the 2nd Annual ACM-SIAM Symposium Discrete Algorithms, San Francisco, California, Jan.1991, p. 241-250.

[LELE87] D. A. Lelewer, D. S. Hirschberg, “Data Compression”, ACM Computing Surveys, Vol. 19, No. 3, Sept. 1987, p. 261-297.