Cellular Data Traffic: Analysis, Models, and Scenarios

By

Xinan Zhou, M.Sc.

A Thesis Submitted to

the Faculty of Graduate Studies and Research

in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Ontario

June 2000

The undersigned hereby recommend to

The Faculty of Graduate Studies and Research

acceptance of the thesis,

Cellular Data Traffic: Analysis, Models, and Scenarios

submitted by

Xinan Zhou, M.Sc.

in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science

_____

Director, School of Computer Science

_____

Thesis Supervisor

Carleton University

June 2000

Abstract

Networks are becoming more and more popular and the traffic over the networks is growing rapidly. Because the network infrastructure is expanding at a quick pace, especially the growth of the Internet, both in size and in the types of applications, the performance issue and dimensioning issue of networks are becoming more critical. It is important to provide a robust and flexible networking environment. To design a system that can deal with these issues well at a reasonable cost, the first step is to characterize the network traffic. In order to get traffic properties that are general it is necessary to study traces of networks in various environments and of different types. A detailed analysis and characterization of network traffic is fundamental to obtain deeper insights into the network system, to fully realize the potential improvements in network performance, and to optimize management of the resources. A lot of work has been done in traffic analysis. The major part is on the Internet traffic. Recently, wireless communication has moved from voice service to data service. Wireless Application Protocol (WAP) makes it possible for commonly used devices such as cell phones, PCs and PDAs to access the Internet. These new wireless data applications produce data that may have different characteristics from those of wired data applications and wireless voice data studied before. In this thesis, we study the data generated by Mobile Browser applications in a cellular network of Bell Mobility in Quebec and Ontario, and compare our results with other results of Internet traffic characterization. We also do some simple performance prediction of the cellular network using Layered Queuing Models (LQM).

Acknowledgements

Special thanks go to my supervisor, Dr. Thomas Kunz, whom I can always count for valuable suggestions and clear guidance. Without his support I may never have finished this thesis. I would also like to thank the members of my committee for their helpful suggestions for organizing the thesis.

Many thanks go to Dr. C. Murray Woodside for his valuable consulting in establishing LQM performance models.

I thank Thomas Barry for providing the trace files, Tauseef A. Israr and Dr. Meng Zhao for interesting advice and discussions.

Sincere thanks will be given to Bell Mobility and the National Science and Engineering Research Council (NSERC) for the financial support during the thesis work.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

In this chapter, we will explain the motivations of the thesis, present the contributions of this work, and finally provide the outline of the thesis.

1.1 Thesis Motivations

Computer networks are growing very fast. They play an important role in industry, in business, in education, and even in our daily life. The importance will surely increase with the time passing by. The continuous trend of growth may bring critical issues for network systems that might not be faced before. One is the performance issue: network systems should provide good quality services even under heavy traffic. Another issue is the dimensioning of the network. Dimensioning is about how to match new networks to existing resources, how to plan for the expansion of the client population and how to upgrade and extend existing networks.

The performance of computer networks includes many factors. The server performance is a critical issue for client/server systems. For example, during peak periods, a server might have to serve more than 3 times as many requests than the average number of requests per second. If the server cannot adequately handle the request traffic, the server will fail to satisfy some requests, resulting in unacceptably slow responses or rejecting them repeatedly. For Web sites, the high percentage of requests for dynamic pages is more likely to make the server CPU a bottleneck. Many Web sites, like sport sites or stock market sites, need to provide dynamic content. It is important to examine the server performance under high CPU loads. When the server is operating near

capacity, there is a tradeoff between average latencies and the percentage of requests rejected: acceptable response time can be guaranteed by rejecting a higher percentage of requests.

The network can also act as a performance bottleneck. Usually, links between a client or a server and the Internet backbone might be the bottleneck at peaks. For a Web site, the Web server will maintain open connections with each client while a request is being processed. If the client is slow, the connection will remain open for a long time. If many slow clients make requests to the Web server at the same time, the performance of the Web server may decrease because the slow clients waste the link.

Nowadays, Internet has become so popular that almost all countries in the world provide the facilities to access the Internet. Of the Internet traffic, WWW composes a heavy part. In 1997, WWW traffic accounted for 70% of the total traffic on the Internet [Choi 1999]. The growth of the Internet is apparent in both size and services. So is the growth of the traffic on the network. Traffic measurement provides first hand data for statistical analysis of the network traffic. Actually, traffic measurement itself is an interesting research area.

Modeling the traffic in general can be very difficult. First it is hard to measure and collect data that characterize the network traffic on users' side. For example, it is difficult to determine how many people visit a Web page and how many applications a user launches during a day. Second it is also difficult to generalize traffic characteristics of a specific network to other networks. Because we cannot prove that a group of users of one network has the same habits and behavior patterns as other groups of network users, and the hardware of the two networks might be different too. So it is important to study as

many different data sets as possible. The study of any new types of data is always meaningful for the validation and modification of the existing conclusions.

Network traffic modeling and forecasting is very important. By predicting the future traffic volume, the number of client requests and size of document retrieved and the requirements for other resources correctly, we can design high efficiency algorithms for caches and the whole system, and make accurate dimensioning decisions.

Though there are huge volumes of references about Internet traffic, little work is found on the cellular data traffic analysis. Newly developed technologies such as Wireless Application Protocol (WAP) make it possible for all kinds of handsets to access the Internet. Mobile Browser that is powered by Phone.com enables Internet on wireless phones and handheld devices. WebCare designed by Bell Mobility [Bell 1999], which is an application running on the Mobile Browser, can provide services like viewing the account balance, paying the bill, etc., on Digital PCS phones. There are many other services available.

WAP Forum initiated WAP [WAP 1998]. Its aim is to help wireless networks offer as many and as high-quality services to clients as wired networks. WAP and WWW, both can enable the clients to browse the Internet. But WWW is designed for the wired network that has wider bandwidth than the wireless network. WAP is designed for the wireless network. The WAP and WWW are so closely related that formats of WAP content and applications and the communication protocols of WAP are all derived from WWW. WAP uses proxy technology to connect between the wireless domain and the WWW. Mobile Browser enables clients to navigate the Internet wirelessly on their PCS phones and handsets. The model of WAP is roughly composed of client, gateway and

WWW [WAP 1998]. Mobile Browser is run on the handset by which the client makes requests. The gateway translates requests from the client so that the WWW can understand them and passes the translated requests to the WWW. The WWW sends the response back to the gateway according to the requests. The Gateway then encodes the response content into compact format and passes it to the client. Because of the limited wireless bandwidth, the reduction of the response content is necessary. The gateway has its own cache to store some information according to a schedule so that some requests from the clients need not be passed to the WWW. This will save time and resources and improve performance. What kind of scheduling (FCFS, FCLS, etc.) will benefit the system most is another interesting research topic. Figure 1.1.1 shows the WAP model. The gateway bridges the wireless and wired networks.

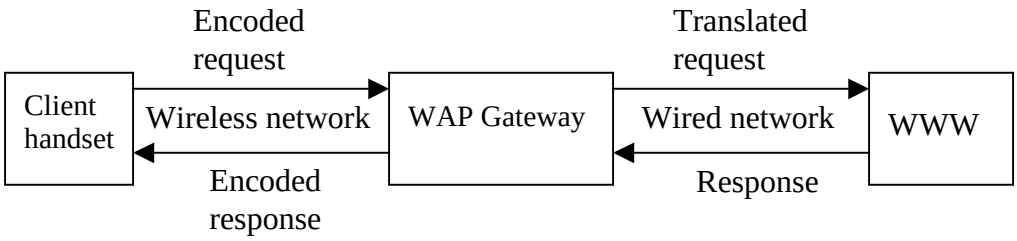

Figure 1.1.1

In this thesis, we will study the Mobile Browser application traffic data that were measured (new traces are kept being recorded continuously) on the cellular network of Bell Mobility in the areas of Quebec and Ontario. The cellular network was formerly designed for voice service. But now it can provide many kinds of data services. The customers initiate the applications on their Digital PCS phones. The application can be

any service provided by Bell Mobility, such as bill payments, stock trading, weather or sports news, and Web site visit. Any application that is being used today with a modem and a phone line can be set up on the Digital PCS phone [Bell 1999]. After dialing and setting up the connection, Web-based applications can be run on the PCS phone wirelessly. During the connection, each client is assigned an IP address. After the application ends or if the application is idle for more than 90 seconds, the IP address may be assigned to another client. We call the IP address client IP address. The server has one unique IP address all the time. But clients are assigned IP addresses dynamically. That is, one device is not related to one unique IP address all the time.

The analysis of the trace files produced from data services on the cellular network of Bell Mobility will provide a solid ground for the optimization of resources of the cellular network and potential expansion and upgrading of the network in the future. Through studying the trace files, we try to answer questions like:

1. When is the busiest time?
2. What is the weekly traffic pattern?
3. Is there a monthly traffic pattern?
4. Is there a traffic growth trend in a certain period?
5. Are there seasonal trends in the traffic flow?
6. Is the traffic self-similar?
7. What are the link utilization or activity factors?
8. What is the distribution of session size?
9. What is the pattern of concurrent sessions?
10. What is the distribution of the session length?

11. Can some properties be predicted with reasonable accuracy?

12. Are the results of our study different from other results, in particular results reported for the WWW?

By using Layered Queuing Models (LQM) and some parameters obtained from the above results plus some assumed parameters, we try to answer the following performance questions:

1. What is the bottleneck of the system?

2. What is the maximum number of clients that the system can serve at the same time?

3. How much the system will be improved if we either increase the CPU speed or add additional CPUs?

4. How does the system behave if we change the values of some parameters?

We have searched many references about data traffic analysis, but did not find one describing cellular data traffic analysis. This work could be the first one on the topic of cellular data traffic analysis and modeling.

1.2 Thesis Contributions

We consider the following as the contributions in this work.

- Analyzing the cellular data trace files and providing some basic characteristics of the cellular data traffic.
- Providing a layered model to predict the traffic and other properties.
- Providing a session model and related features.
- Deriving some parameters from the session model.
- Comparing the results in this thesis with previously published results.

- Studying the abstract WAP system by two LQM models.

  Part of the work is presented in two workshop papers: [Kunz 2000] and [Omar 2000].

1.3 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 provides basic material about the network and traffic analysis, and overviews related work on Internet traffic analysis, especially on WWW traffic. Chapter 3 introduces the characteristics of our trace files and the environment of the cellular network where the trace files are recorded. The results obtained by analyzing the trace files are provided too. A layered prediction model is devised and shown to be reasonably accurate by several examples. Chapter 4 gives a session model and shows some results based on the session model. The session characteristics in a month are described for July 1999 to save the space. The Results in other months are in Appendix A. Relationships between some properties are found and discussed. Some attempts are made to fit statistical distributions onto the results. In Chapter 5, our results are compared with corresponding results in the literature. We investigate the reasons for the similarities and differences. Chapter 6 gives two simple LQM models of the system and some performance studies are carried out with these two LQM models. Chapter 7 summarizes our findings and discusses potential future work. In Appendix A, we provide session analysis results for August, September, October, November and December, which are omitted in Chapter 4. We also present information about maximum concurrent sessions from July 1, 1999 to Dec. 31, 1999.

Chapter 2: Background and Related Work

In this chapter, we will give a brief introduction to data networks and discuss related work of Internet traffic analysis, especially WWW traffic analysis.

2.1 Introduction to Data Networks

Data communication has a history that is almost as long as the history of human beings. In ancient times people used all kinds of ways to pass information such as sounds, hands, flags, smokes and mirrors. In modern times telegraphy is considered as the beginning of data communications. From the 1920s to the 1960s telegraphy was the major way to provide data communication services. When computer networks appeared, people found it a more powerful and suitable way to provide data communication services. As computer technology developed, packet switching was found more suitable for data communication than circuit switching. For example, X.25 can provide robust data transfer services.

The network is designed in a layer-structured way and a higher layer is built upon a lower layer. The lower layer provides necessary services only to the adjacent higher layer. Each layer has its own layer protocol. International Standards Organization (ISO) proposed a reference model for Open System Interconnection that includes seven conceptual layers. In the increasing order of the layer number the seven layers are physical hardware layer, data link layer, network layer, transport layer, session layer, presentation layer and application layer (X.25 is an example implementation of the ISO model). The physical layer is responsible for transmitting a stream of electrical digital data bits over a physical channel. The data link layer defines the format of frames and

transfers data in a unit of frame. The network layer specifies how a start host sends packets to a destination host. This layer concerns the destination addressing and routing. The transport layer is responsible for the end-to-end transport reliability between the source end and the destination end by making sure no errors occurred in the middle of communication. The session layer provides the necessary functionality that is needed by application programs. The presentation layer provides all the necessary ways to transform the data of the end users. The application layer is for application programs that are commonly provided by the networks like email and file transfer [Comer 1988]. In real life TCP/IP is widely used, not the ISO model. TCP/IP is based on a model with only five layers: physical layer, network interface layer, internet layer, transport layer and application layer. It is also often called the Internet model. The ARPANET was developed by the United States Defense Advanced Research Projects Agency (the agency was formerly known as ARPA, not DARPA) to interconnect many of its computers in a distributed way. It is the beginning of Internet. The ARPANET is an example of the implementation of the Internet model. The Internet model structure is different from the ISO model structure. The Internet model combines application layer, presentation layer and session layer into only one application layer and adds an internet layer between the network layer and the transport layer. The application layer in the Internet model provides all the services that are provided by application layer, presentation layer and session layer in the ISO model. The internet layer concerns issues such as error checking, handling incoming datagrams, data validity checking, header detecting and data routing. The internet layer uses protocols such as Internet Protocol (IP) and the Internet Control Message Protocol (ICMP). TCP and UDP are at the transport layer. The application layer

in the Internet model usually provides functionality such as Telnet, FTP, the Simple Mail Transfer Protocol (SMTP), the Domain Name Server (DNS) Protocol, and World Wide Web (WWW). The Internet is formed by a group of separate networks worldwide that are connected through repeaters, bridges, routers, and gateways. A repeater just passes bits from one network to another at the physical layer, and is transparent to the higher layers. A bridge combines two networks at the data link layer. A router connects isolated networks at the network layer. A gateway bridges different subnetworks that have different transport protocols [Tanenbaum 1996]. Each host on the Internet has a unique address called IP Address that is 32 bits long. Each IP address corresponds to a name because people would like to remember machines by names not by a long string of numbers. DNS is used to map names of machines into their corresponding IP addresses respectively. More concepts and detailed explanations about IP, TCP, UDP, DNS, and WWW, etc. can be found in [Tanenbaum 1996] and [Comer 1988].

2.2 Brief Introduction to Wireless Networks

Radio techniques started the wireless communication. They evolved into ocean vessel radio, vehicular mobile radio and aircraft radio. But the moving distance is limited. AMPS is the result of extensive research by Bell Labs in 1960s and 1970s [Redl 1998]. Since then the cellular idea has been widely accepted by the wireless world. The AMPS system is different from previous mobile radio systems in that it interfaces with PSTN. AMPS works by organizing a group of adjoining cells, each cell is managed by a base station (BS) and has the capacity of handoff when a client moves through the cell to another. The base stations are connected to public switch telephone networks (PSTN)

through the mobile switching centers (MSCs) by land links (see Figure 2.2.1). The client communicates with the BS that manages the cell the client is in wirelessly. AMPS is the first generation cellular system that is analogue. Europeans came up with a TDMA system, GSM, the second generation cellular system [Redl 1998]. GSM is a solution to transfer from analog to digital. The third generation is at the research stage now.

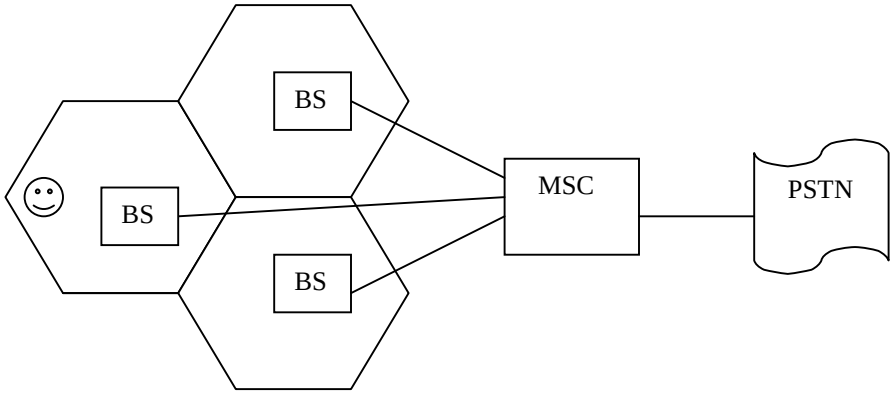

Figure 2.2.1

Before 1992 all cellular systems were analog. In analog cellular systems, the speech signal, a continuous smooth analog waveform, is encoded directly onto the carrier [Lee 1998]. Since 1992, most newly deployed cellular systems have been digital, this means that a sequence of 1s and 0s is transmitted rather than the analog signal. Digital systems are generally considered to be better than analog system. The problem with analog systems is that they are too sensitive to interference. For example, a small amount of change on the input signal can result in a big change to the output signal. Digital systems are less interference sensitive, and promise higher capacity for the same amount of radio spectrum than analog systems.

FDMA, TDMA and CDMA [Gibson 1999] are ways of dividing up the radio spectrum so that a number of users can talk at the same time. It is commonly agreed that

TDMA and CDMA are much more efficient than FDMA. But arguments about whether CDMA was better than TDMA or whether the Qualcomm IS-95 cellular standard (now known as CDMAone) [Lee 1998] was better than the GSM cellular standard are still on. Jacobs and Viterbi claimed in 1991 that the CDMA cellular system had a capacity that was 20 or so times greater than any other cellular system in existence [Gibson 1999]. However, at the present time CDMA systems provide a capacity probably around 30% greater than TDMA systems, far below what Jacobs claimed [Gibson 1999]. And CDMA system components cost more than TDMA components.

In the design of wireless networks, requirements of voice and data services are different. For voice services the delay should be minimized as much as possible. If the delay is longer than 100 ms, the listener can notice the delay and feel uncomfortable [Pahlahvan 1994], and the voice service becomes unacceptable. But the delay in a data network is generally acceptable to the data user. Though voice services cannot bear 100 ms delay, they can tolerate a packet loss rate that can be as high as 0.01 [Pahlahvan 1994]. On the contrary, data services do not permit any packet loss. No one wants to download a file with errors since it is useless.

Usually there are two types of wireless data networks that serve different purposes: Wide-Area Systems and Local-Area Systems. Wide area wireless data systems are designed to provide high mobility, wide area coverage, but the data transmission speed is low. They typically provide various short-message applications such as notice of electronic mail and performing transaction services. Local area wireless data systems are often designed to support a limited number of users in a small area, but the data

transmission speed is high. They typically support various local applications such as long file transfers or printing tasks that require high speed [Garg 1997].

Newly developed technologies such as WAP make it possible for devices like cell phones, PCs and PDAs to access the Internet. WAP has a layered architecture that makes WAP scaleable and extensible. WAP exists on top of GSM, CDMA, CDPD, etc.

2.3 Performance Issues

Performance is critical for client/server type systems. Many researchers are developing new techniques that can improve performance. Results obtained by studying traces can provide guidelines for performance studies.

For the WAP model, the time needed to complete a request include the processing time at client, gateway, WWW and any devices in between, and queuing time at gateway, WWW and intermediate routers in between. To guarantee a quick response to a request, it is important to have enough network resources (bandwidth and buffer sizes) to deal with the peak load in the network. If the network resources are limited it is necessary to control the amount of data that is sent into the network to avoid congestion. Through caching, the number of client requests sent to distant server can be reduced. But for the wired network part, we cannot control the traffic volume because of its being shared by everyone who can access the Internet. The correct match of network and server resources to anticipated demands and the careful design of caching schemes can improve the quality of the services provided by the systems.

2.4 Related Work

Due to the fast expansion of the networks, especially the Internet and newly available digital data services by all kinds of handsets, total traffic has increased enormously. Many efforts focus on network measurement and performance and dimensioning studies of networks. There exists a lot of literature about Internet behavior in recent years.

2.4.1 Characterizations of Internet Traffic

The first step to characterize the network traffic is to measure traffic flow on the networks. Depending on what aspects to study and what information is needed, people can choose to measure traffic on the client side, on the proxy side, on the server side, or a combination of these. The measurement can be at the packet level, at the TCP level, or at the application level. Decision on which level to use may be based on factors such as which level can provide the necessary information and which level is possible technically and legally.

More information can be obtained by measuring on the client side, but often it is difficult. For measuring WWW traffic on the client side, the difficulty is that it is hard to modify WWW browsers in order to record user behavior related data. Moreover we cannot guarantee that the characteristics of the user's behavior that has been studied are the same as other users.

[Catledge 1995] studied client behavior in the WWW in 1994. They modified Mosaic into Xmosaic and recorded three-week traces at the Georgia Institute of Technology. The number of clients was 107. The results revealed the way people used Mosaic that was the most often used browser at that time. By analyzing the traces they found people liked to search back and forth from the initial page. This provided a solid

base for behavior model building. In the same year, the Boston University Oceans Group conducted another study on a larger scale that had over 600 users and lasted over six months. They also used Xmosaic. The obtained traces were used widely. [Cunha 1995] and [Crovella 1997] found that the distributions of transmission times and document sizes versus number of requests were Pareto. The distribution of document popularity follows Zipf's distribution. [Crovella 1995] and [Crovella 1997a] demonstrated that WWW traffic had the nature of self-similarity.

Many researchers like to study WWW traffic by analyzing proxy traffic traces, because there are more proxy traces available than client traces. But many of these studies focus on improving caching algorithms only. [Leland 1994] studied the traces collected on several Ethernet LAN's at the Bellcore Morristown Research and Engineering Center from August 1989 to February 1992. They reported self-similarity of the traffic and gave $H = 0.8$, after analyzing a sample of 27 hour (sampling time interval is 10 ms) by variance method, R/S plot, and periodogram plot. [Paxson 1995] analyzed 24 traces of wide-area TCP traffic. The traces they used were from Bellcore, UK-US, coNCert, etc. They reported self-similarity of the network traffic, but did not give the H value (H is a Hurst parameter used to describe the degree of self-similarity). [Abdulla 1997a] did spectral analysis to the proxy traces, and found the bursty behavior of the traffic and predictable daily and weekly periods. [Abdulla 1997c] conducted an extensive analysis of proxy logs from ten sources, and claimed finding nine invariants. They also revealed that the Hurst parameter varies between 0.59 and 0.94 across all proxy traces and added this finding as another invariant. [Thompson 1997] measured traffic on the OC3 link of the network at the U.S. East Coast. The daily traffic pattern and weekly

traffic pattern are obvious. The rate of traffic flow ranges from 10 Megabits/sec to 55 Megabits/sec. They also described proportions of many components in the traffic such as WWW, DNS, SMTP, FTP, and TELNET. [Mah 1997] used the tcpdump-capture utility to record TCP/IP packet headers on a shared 10 Mbps Ethernet in the Computer Science Division at the U. of California at Berkeley in late 1995. The results showed that HTTP requests exhibit a bimodal distribution and the sizes of HTTP replies are heavy-tailed. [Arlitt 1999b] studied the effects that cable modems have on proxy workloads. The data were collected at an Internet Service Provider (ISP) from January 3, 1997 to May 31, 1997. They found that users were more likely to download extremely large files if they had faster access speed.

Another important way to study Internet traffic is to measure the traffic flow on the server side and analyze the collected traces. There are many published papers about WWW server studies. [Mogul 1995] analyzed traces obtained from the Californian congressional election server set up by DEC on November 9, 1994. He found that the inter-arrival time of requests did not fit a pure Poisson process. [Arlitt 1996] conducted an extensive analytical study, and claimed ten invariants that were consistent with results of many other researchers. The periods of the traces that he used in his research vary from one week to one year. [Almeida 1998] did analysis on access logs from NCSA, SDSC, EPA, and BU during the fall of 1996. They found that popularity for documents served by Web sites and sequences of requests from clients follow Zipf's Law. [Arlitt 1999a] gave a detailed workload characterization study of the 1998 World Cup Web site and compared them to those obtained from other Web server workloads. The data were measured from the 1998 World Cup Web site for a period of three months. The authors

revealed that caching at Web clients, proxies and within the network would change the workload that could be seen by Web servers.

2.4.2 Web Traffic Models

SPECweb96 [SPEC 1996] and WebStone [Silicon 1996] are commonly used benchmarks to generate Web server access patterns and to analyze Web server performance issues. WebStone issues as many requests as possible from synchronous clients, and thus can measure the maximum request rate that a Web server can sustain. WebStone is merely used to test the system capacity, and it is not capable of characterizing the burstiness, trends, interdependencies and seasonal behavior of requests. SPECweb96 has the same problems as WebStone, but it can characterize delays between client requests.

[Mathur 1996] constructed an empirical workload model fitting time-varying trace data. The trace is modeled as a piecewise independent stochastic process. This model is appropriate to generate trace data that can be considered as a time series. But the key assumption in the model is not valid in real cases.

[Lam 1996] presented a realistic model for PCS. The model was claimed to capture complex human behaviors. Based on the model the authors developed Pleiades, a discrete event simulator. But the validation of Pleiades on large-scale tests is not provided.

[Barford 1997] developed a tool called SURGE. Based on statistical distributions in WWW server usage SURGE can generate traffic that is very close to the real world. In SURGE, the distributions of parameters such as file size, locality, etc., are used as the inputs. These distributions of the parameters can be derived from experiments. SURGE

can simulate Web sites very well statistically. SURGE can also mimic the behavior of real clients. The workload generated by SURGE is more close to the real world than the workload generated by SPECweb96 and WebStone. SURGE can keep a larger number of open connections at the same time and can generate self-similar network traffic at heavy loads.

[Mah 1997] provided an empirical model that could mimic WWW network applications. But the server selection distribution in the model is not changeable and the relationships between the different model components are not investigated. [Mah 1998] developed an IP Benchmark (IPB). IPB can synthetically generate traffic to simulate the network activity of common Internet applications and thus be used to measure HTTP performance. But IPB does not include enough traffic types and updated models of existing applications. [Almeida 1998] provided a model, called the Wisconsin Proxy Benchmark (WPB). This model can simulate the request streams according to the temporal locality patterns that are common to Web proxy servers.

[Iyengar 1998] developed a tool called Flintstone. This model uses statistical methods to isolate and characterize the trends, interdependencies, seasonal behavior and noise in the access patterns. Flintstone provides an effective approach for predicting peak request rates for analyzing and characterizing Web access patterns. Flintstone can also generate realistic workloads for benchmarking Web servers. SPECweb96 and WebStone cannot generate inter-request times that are based on actual Web request data. In contrast, Flintstone generates request data that reflects the trends, burstiness, interdependencies and seasonal behavior that occur in real situations. Flintstone also generates request traffic that can be scaled to different arrival rates. Later, the authors extended the

methodology of Flintstone by introducing the logARIMA, or eARIMA, process. The extended model can incorporate heavy tailed distributions together with a set of time series processes, in addition to addressing the trends, interdependencies, seasonal behavior and noise of non-stationary time series data.

[Choi 1999] presented a behavior model of Web traffic. It defines a new unit, a Web-request, different from a traditional Web page. It can simulate detailed dynamics of TCP/IP as well as HTTP.

2.5 Summary

In this chapter, we briefly introduced data and wireless networks. Then we discussed some performance issues, and reviewed related work about Internet traffic analysis and different benchmarks. Benchmarks are not the primary concern in this work. We will focus on the Internet traffic characterizations. However, the results reported in this thesis could be used to design appropriate benchmarks for WAP applications.

Chapter 3: Analytical Results of Trace Files

In this chapter, we will introduce the trace files used in the work and give basic analytical results. We also put forward a layered forecasting model and show several examples of the application of the model. In order to express the results more clearly we give the following definitions.

- **Forward packet (f-packet)**: a packet that is transferred from the server to the client.
- **Backward packet (b-packet)**: a packet that is transferred from the client to the server.
- **Client IP address**: an IP address that is assigned to a client during a session.
- **numIPe**: number of unique client IP addresses in forward packets.
- **numIPs**: number of unique client IP addresses in backward packets.
- **numIP**: number of unique client IP addresses in all packets.
- **t-traf**: total traffic, sum of forward traffic and backward traffic.
- **f-traf**: forward traffic, sum of all forward packet sizes.
- **b-traf**: backward traffic, sum of all backward packet sizes.
- **t-packet**: total number of packets, including forward packets and backward packets.
- **f-packet**: total number of forward packets.
- **b-packet**: total number of backward packets.
- **t-a-p**: average packet size, over all forward and backward packets.
- **f-a-p**: average forward packet size.
- **b-a-p**: average backward packet size.

3.1 Trace Files

The trace files used in this work are obtained from Bell Mobility's cellular network, located in Quebec and Ontario. The cellular network consists of base stations and interworking functions (IWFs). IWF functions like the MSC in Chapter 2. The measured point is at the Gateway server that connects the Internet and IWFs of Bell Mobility's network, see Figure 3.1.1. TCPDUMP is used to record the UDP packets to and from the Gateway server that has the unique IP address 161.216.17.21. The services provided by the Bell Mobility cellular network are client-server type. The clients make queries to the server, and the server provides answers to the clients.

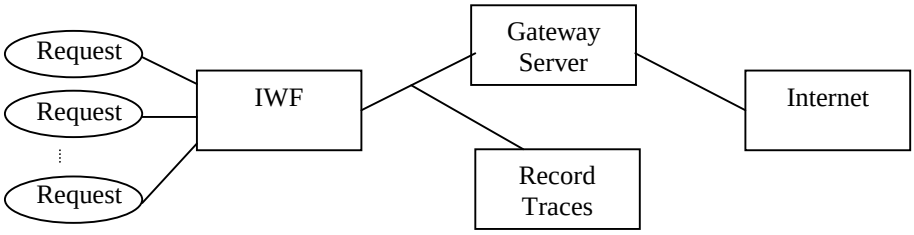

Figure 3.1.1

The trace files are recorded continuously. But sometimes, the trace files are not recorded for failures of equipment. In July 1999, trace files of the whole month are available. In August 1999, trace files were collected from August 1 to August 24. In September, trace files were collected from Sept. 7 to Sept. 30. Two days are special. In Sept. 7, only data from 4:14PM to 11:59PM was collected. On Sept. 13, only the data from 00:00AM to 9:07AM was collected, there were no data records between 9:08AM to 11:50PM, and one packet was recorded at 11:50PM. It is highly unlikely that there was no traffic during such a long period, so we assume that some failure happened at that

time. Trace files in October, November, and December, are available without a hole. The information in the trace file includes the IP address of the sender, the timestamp that is created at the collector, the size of the UDP packet, the IP address of the receiver. Because we know the IP address of the server, we can distinguish forward traffic from backward traffic. We can characterize the traffic over several time scales, e.g., 1 second, 1 minute, 1 hour, 24 hours, 7 days and 1 month. All the trace files are pre-processed so that they are more easy to read and more easy to further process. Each line in the pre-processed trace files has the same format. Each field in the line is separated by a space from another field. An example is as follows:

9 7 1999 16 14 25.33 207.38.2.5 8502 161.216.17.21 1908 12

The first field is month, 9 means September. The second field is day, 7 means the 7th day in the month. The third field is year, which is 1999 here. The fourth field is hour of the day, and it is 16 here (i.e. 4PM). The fifth field is minute. The sixth field is second. The seventh field is the IP address of the sender. The eighth field is the port used by the sender. The ninth field is the IP address of the receiver. The tenth field is the port number used by the receiver. The number 12 in last field means that the payload in the UDP packet is 12 bytes. This number does not include the UDP header, which are 8 bytes, and IP header, which are 20 bytes. So the packet size for each UDP packet should add 28 bytes to the payload. For the example above, the size of the packet is 40 bytes.

The sequence of numbers, 161.216.17.21, is one unique IP address assigned to the server. In the above example, the packet is transferred from the client to the server, so it is a backward packet. The trace collection is continuing every day. We cannot update the data all the time. We only use the trace files from July 1, 1999 to December 31, 1999 in

this work. Some update of the results may be needed when we include new trace files for analysis.

3.2 Statistical Results of the Trace Files

In this section, we will provide some statistical properties of the trace files and plot the traffic in different ways. These will give us a global view of the traffic during the above period.

The inter-arrival time means the time interval between the timestamps of two consecutive packets in the same direction. The statistical results are shown in Table 3.2.1. Some useful information can be deduced from the table. First, there are more backward packets than forward packets. Second, the average backward packet size is smaller than the forward packet size. Third, backward packet sizes are closer to the average backward packet size than forward packet sizes are to the average forward packet size (smaller variance).

Table 3.2.1 Statistical Results of the Trace Files

	Maximum	Minimum	Total number	Mean	Variance
All Packets	1485 bytes	30 bytes	6884602 packets	94.09	100.8
Backward packets	1105 bytes	30 bytes	4366001 packets	52.72	15.84
Forward Packets	1485 bytes	31 bytes	2518601 packets	163.73	141.82
Backward inter Arrival time	53001.07 seconds	0 seconds	4365830	3.33	68.32
Forward inter Arrival time	19786.61 seconds	0 seconds	2518430	5.75	84.84

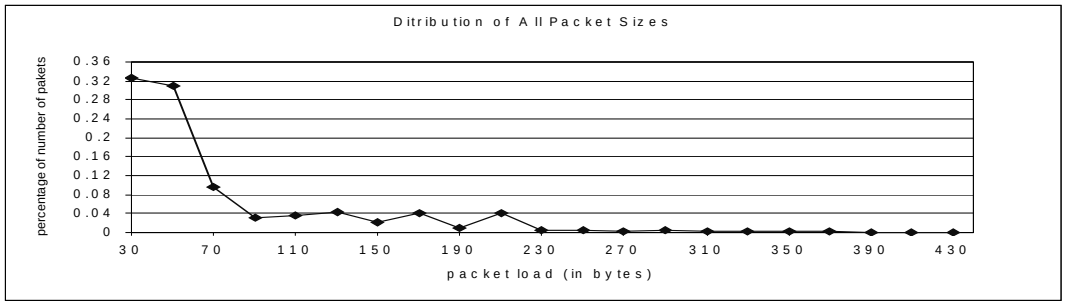

Figure 3.2.1

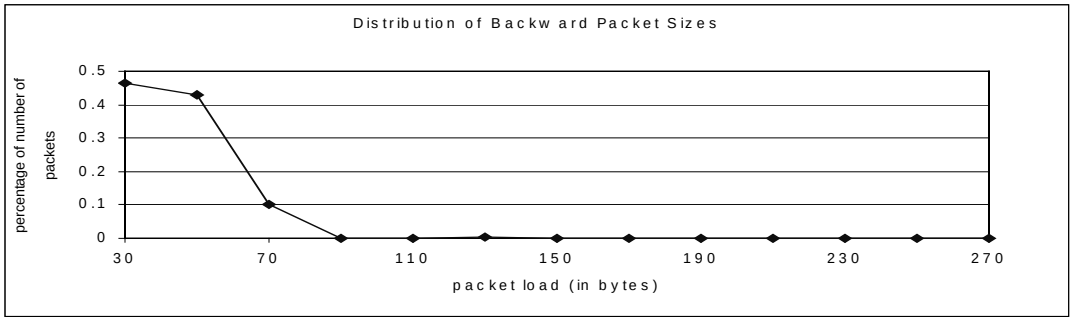

Figure 3.2.2

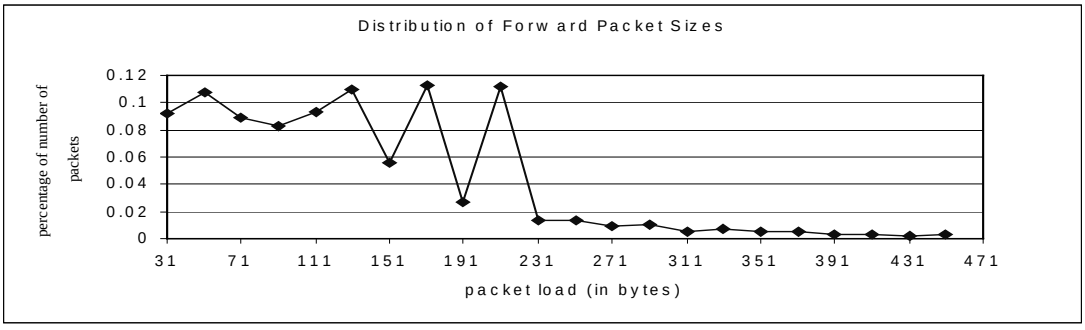

Figure 3.2.3

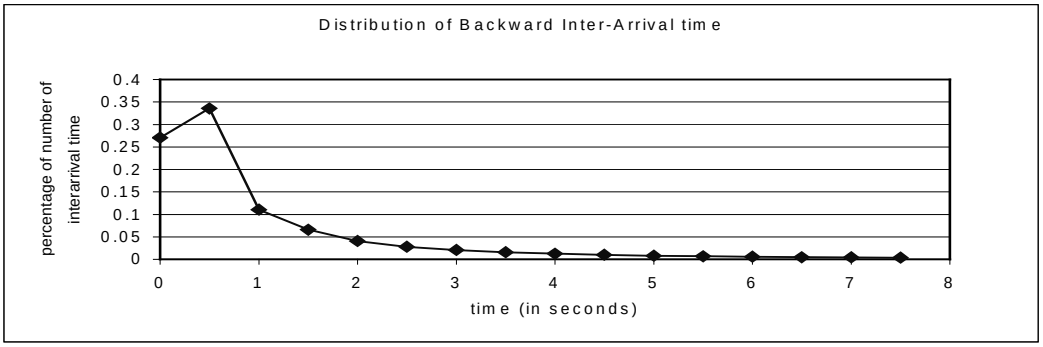

Figure 3.2.4

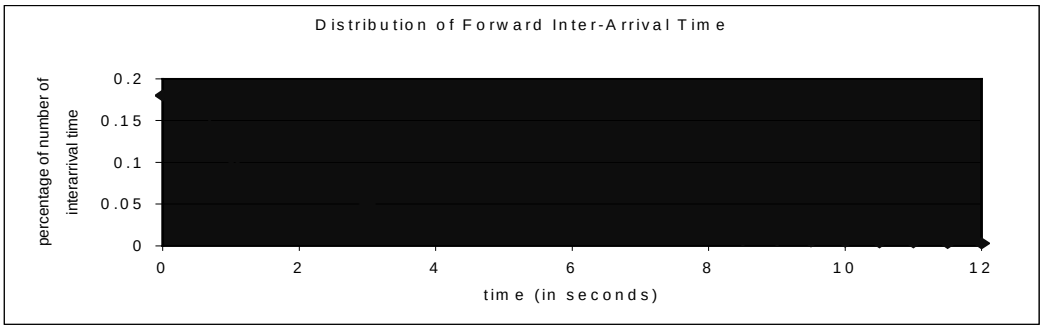

Figure 3.2.5

The distributions of packet size and inter-arrival time are shown in Figures 3.2.1 to 3.2.5. The backward inter-arrival time distribution is not well fitted by an exponential function (commonly used in analytical performance studies). We only use points within the range of 0.5 seconds to 10 seconds. By assuming the fit function as $e^{(at + c)}$ and manipulating natural log, we obtain $a = -0.4415$, $c = -2.1755$ with least-square regression. The mean of error is *0.0173*. The variance is *0.0542*. If we use function $b*t^k$ *to fit*, we get $b = 0.1233$, $k = -1.6859$. The mean of error is *0.0045*. The variance is *0.013*. The two functions are not particularly good fits for the range of 0.5 seconds to 5 seconds. But the latter is better. The fact that the inter-arrival time of backward packets is not well fit by an exponential function implies that the poisson process is not suitable to describe

the arrival process of backward packets, which traditionally is considered to follow a poisson distribution.

For all packets, the number of packets in the range of 30 bytes to 50 bytes accounts for a majority of packets, 63.8%, but account for only 26.9% to 40.5% of total traffic volume. Similar conclusions can be obtained for backward packets and forward packets. That means, though the number of large size packets is small, they account for a large percentage of traffic that cannot be ignored.

The traffic pattern of July 1, 1999 to December 31, 1999 is shown in Figure 3.2.6. August 25, 1999 to September 6, 1999 is the hole in the trace files. We can see that traffic volume has a sharp increase around Oct. 12, 1999. This may be caused by the introduction of new services that trigger more customers to use the network more frequently. If we describe the traffic by the number of packets instead of bytes, we get a similar pattern (see Figure 3.2.7).

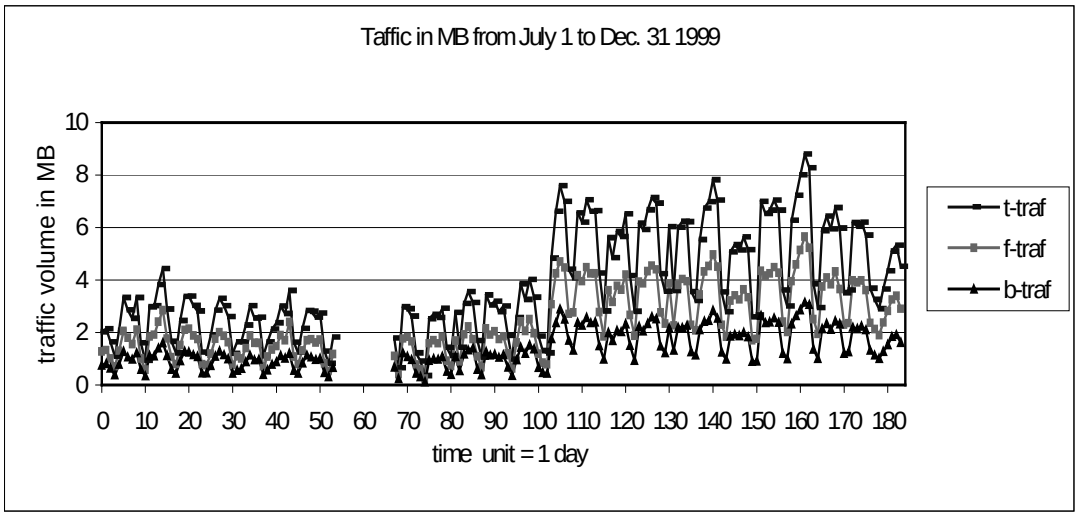

Figure 3.2.6

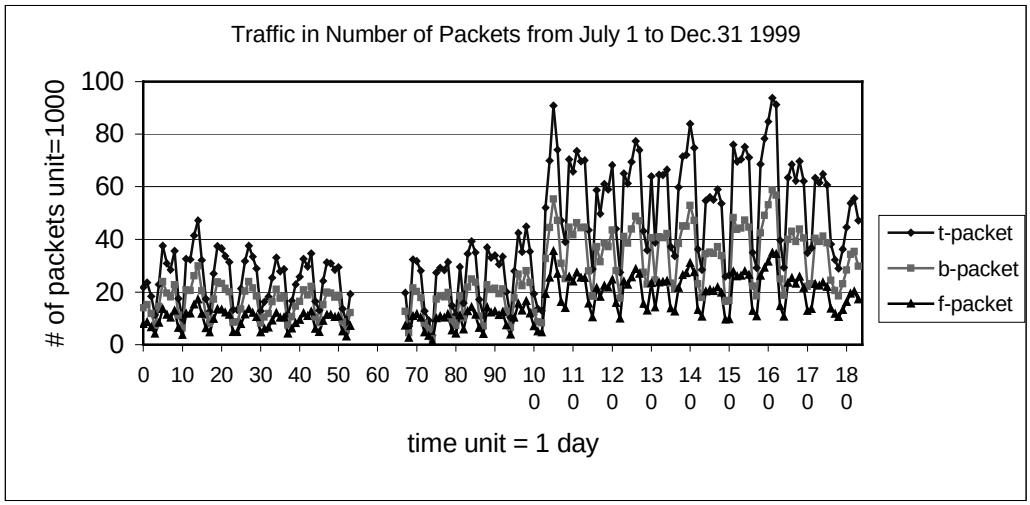

Figure 3.2.7

3.3 Traffic Invariants

One aim of trace file analysis is to find traffic invariants that can provide sound foundation for traffic forecasting. Traffic volume varies with day, but average packet sizes of each day are constants, see Figure 3.3.1. We also find two ratios of each day are constant. For each day, we define

r-pk = number of backward packets / number of forward packets,

r-vlm = forward traffic volume / backward traffic volume.

As shown in Figure 3.3.2, r-pk and r-vlm seem to be constant over time. More surprisingly, the average of the r-pk series is 1.74667, and the average of the r-vlm series is 1.74706. They are almost equal, around 1.75, which is coincidental.

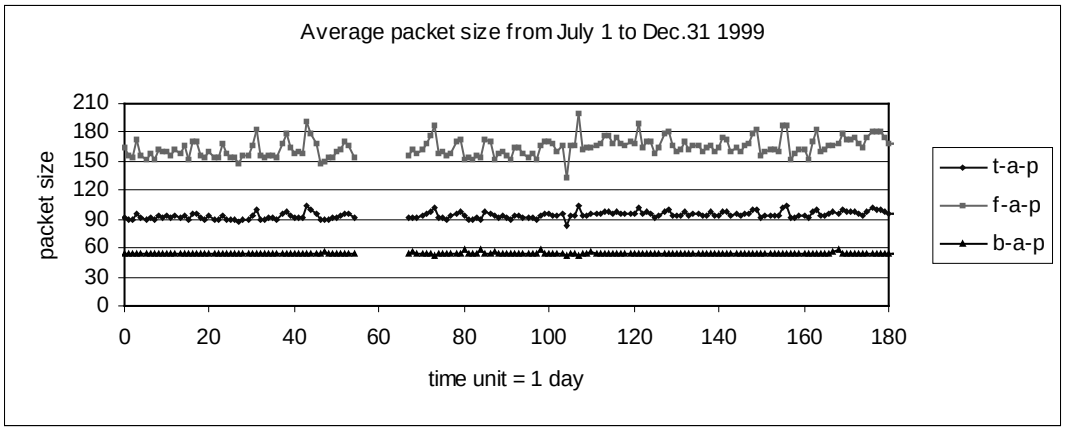

Figure 3.3.1

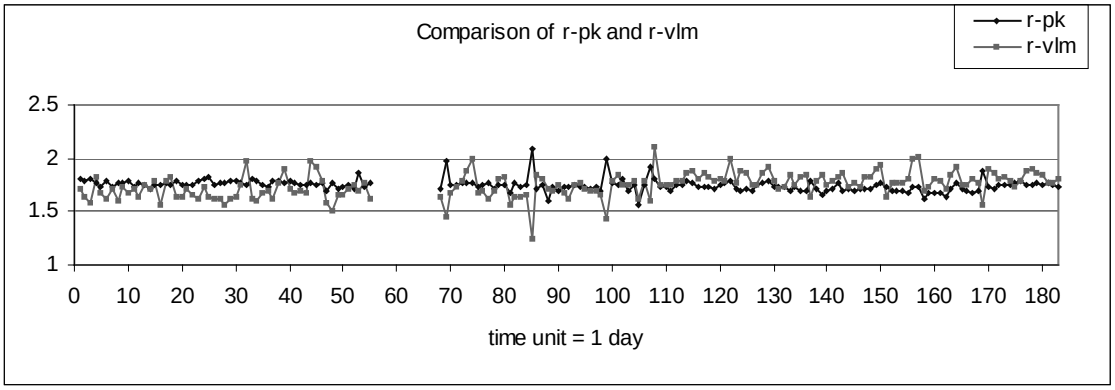

Figure 3.3.2

From Figure 3.3.3, we can see that the number of different client IP addresses per day is linearly increasing with time (especially to the right part of the hole). We claim the rate of increase is a constant. The number of unique client IP addresses in backward packets is equal to the number of unique client IP addresses in forward packets for most of the days.

For few days the two numbers are different, but the difference is small. There are more times that the first number is greater than the second number.

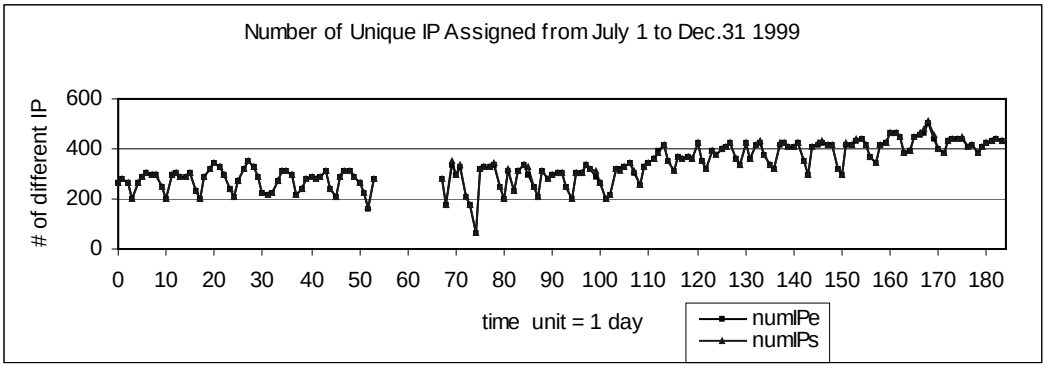

Figure 3.3.3

3.4 Analytical Results

Figure 3.4.1 shows the average daily traffic pattern over the period of July 1, 1999 to Dec. 31, 1999. We can see that traffic from 2:00AM to 5:00AM is low and traffic from 11:00AM to 4:00PM is high. That means people use the service more around noon than around early morning. As shown in Figure 3.4.2, there is a peak every 24 hours. The peaks oscillate periodically. Low peaks appear every 168 hours. They are weekly and daily periods. The traffic is low at weekends and high on workdays, which indicates that people use the service less on weekends than on weekdays.

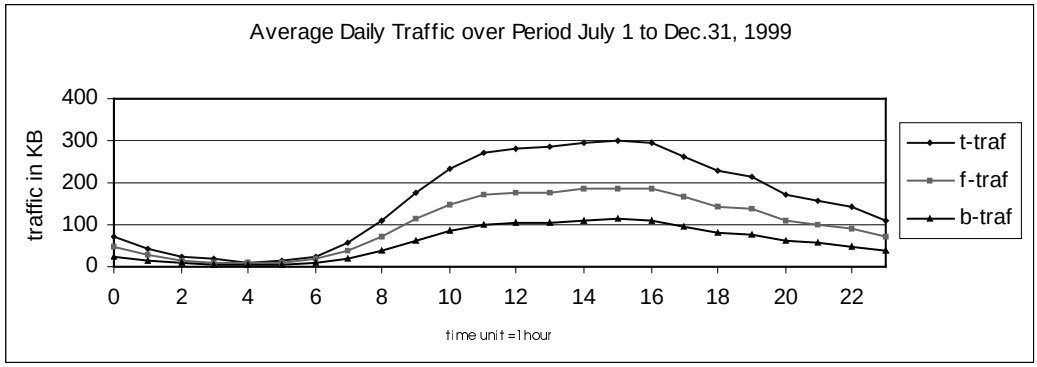

Figure 3.4.1

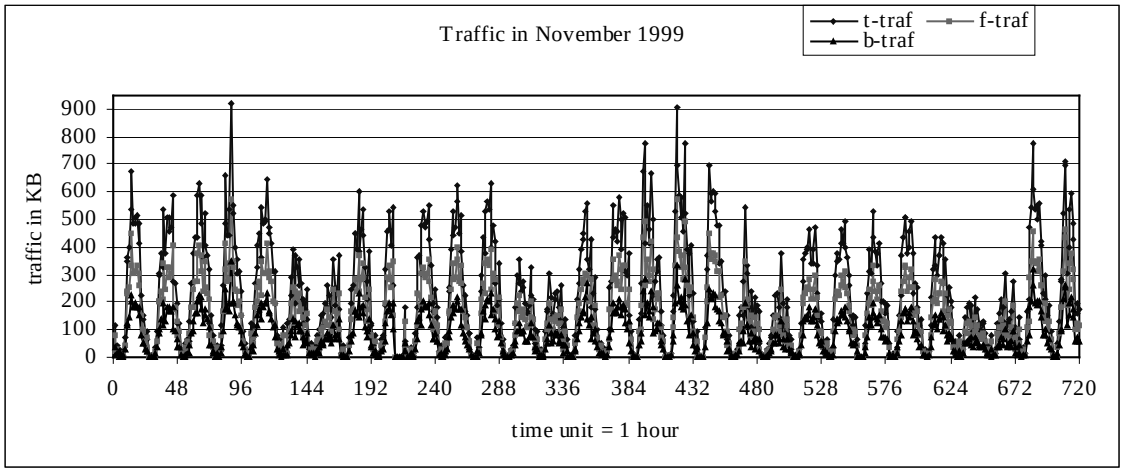

Figure 3.4.2

Next we perform spectral analysis of trace files. To investigate if there exists a monthly period, we should use the data for at least two months. There are no holes in Nov. and Dec., and the trend is almost horizontal, so we first do the spectral analysis on the trace from Nov. to Dec. 1999, and show half day, day, half week, week, month periods. In Figure 3.4.3, the sampling unit is 1 day. The day and half-day periods cannot be revealed (Nyquist sampling interval). The week period is the main component, *T2 = 1/0.140625 (days) = 7.11 (days).* Month period is not very obvious, *T3 =*

1/0.03125(days) = 32 (days). Half week period is not very obvious, *T0 = 1/0.28125 (days) = 3.56 (days).* In Figure 3.4.4, the sampling unit is 1 hour, the day period is the main component, *T1 = 1/0.041504 (hours) = 24.094 (hours).* The half day period is not very obvious, *T = 1/0.083496 = 11.9766(hours).* The week period is not very obvious, *T2 = 1/0.005859 (hours) = 170.6776 (hours) = 7.11 (days).*

We also perform spectral analysis on data shown in Figure 3.3.2 and Figure 3.3.3, and find that the number of client IP address and r-vlm give an obvious weekly period, but r-trans does not.

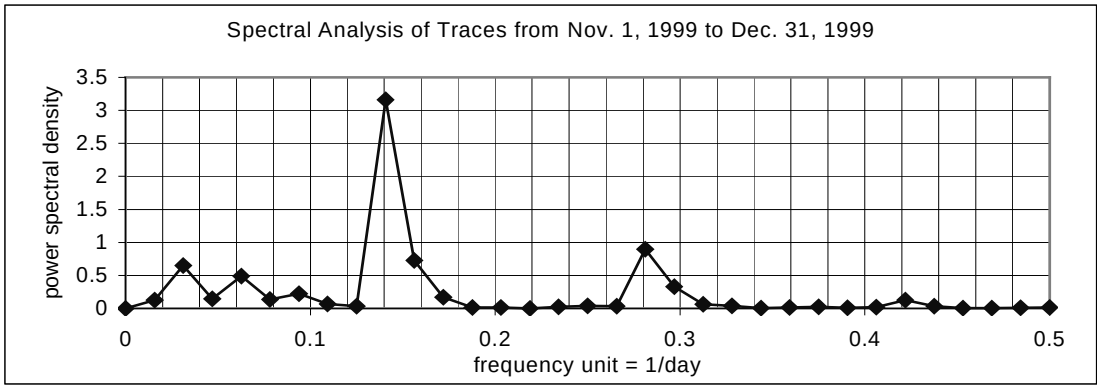

Figure 3.4.3

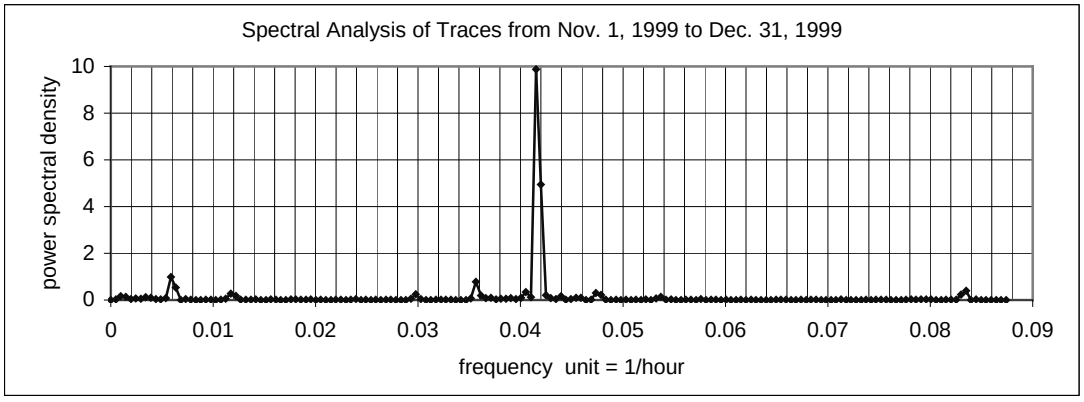

Figure 3.4.4

To investigate the self-similarity of traffic, we use Absolute Value Method (AVM) and Variance Method (VM) [Taqqu 1997] to calculate H values of our traces. We did this laborious work for each day trace. For each day trace, we define low traffic part (00:00AM to 11:00AM for the first calculation, 1:00AM to 6:00AM for the second calculation), and high traffic part (11:00AM to 10:00PM for the first calculation, 10:00AM to 3:00PM for the second calculation), and calculate H values for both parts. The reason that we divide a day traffic into high and low part is that we want to find if the traffic volume is a fact that will influence the degree of self-similarity. We found H values are from 0.70 to 0.92. We also calculate H values for the traffic series from July 1, 1999 to August 24, 1999, $H = 0.84$, and the traffic series from September 14, 1999 to December 31, 1999, $H = 0.86$. We start from September 14, because we find that the traces on September 7 and 13 are only half day, not continuously. H is the Hurst parameter that refers the degree of self-similarity. For AVM, H = $1 + r$; for VM, H = $1 + r/2$, where r is the slope of the line plotted by AVM and VM. We use least square method to obtain the slope of lines here. H should be within 0.5-1.0. The closer H is to 1 the higher is the degree of self-similarity. More details can be found in [Leland 1994]. As examples, we only give the absolute value plot and variance plot for July 1 1999, see Figure 3.4.5 to Figure 3.4.8.

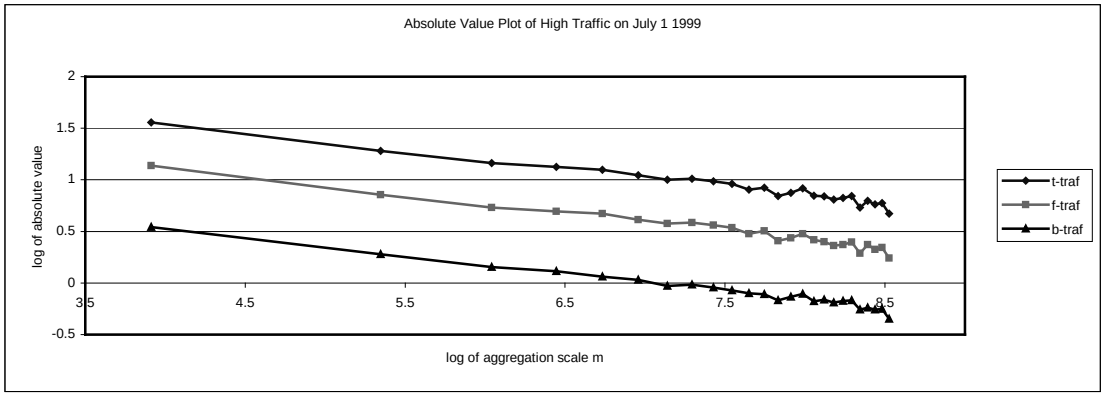

Figure 3.4.5

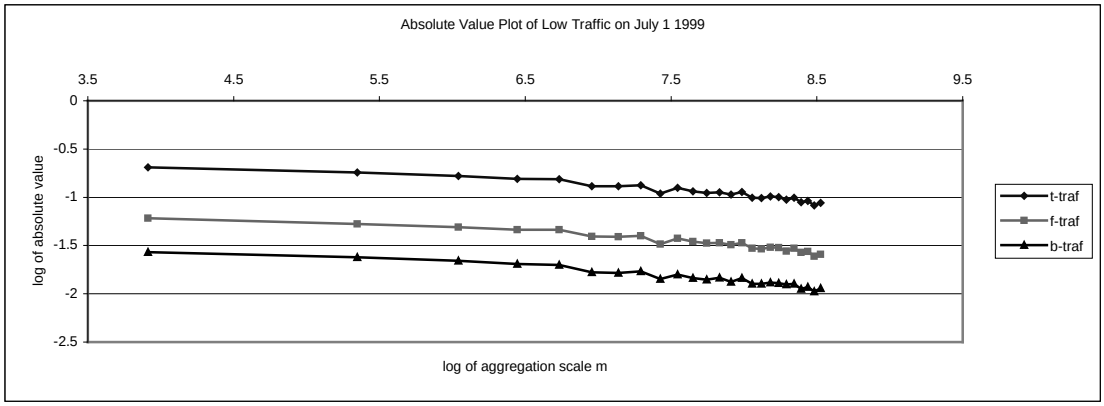

Figure 3.4.6

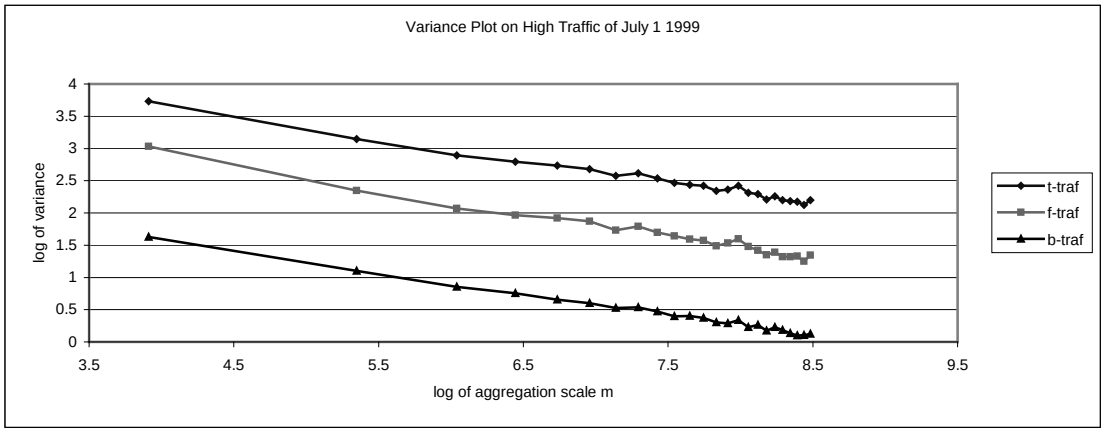

Figure 3.4.7

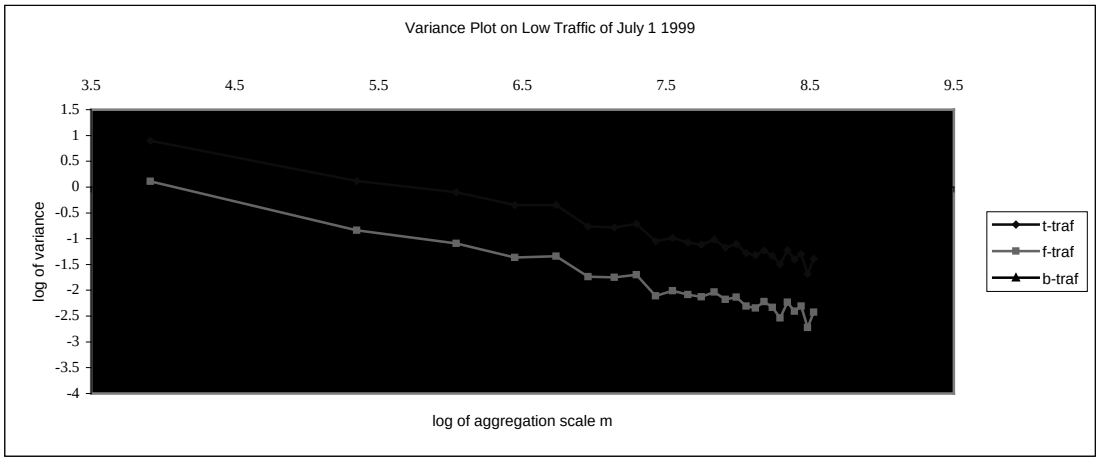

Figure 3.4.8

3.5 A Layered Forecasting Model

One aim of studying trace files is to predict the traffic and other properties in the future. There are many books and papers related to prediction of time series, which is a big topic by itself. The commonly used way is to provide a formula according to the existing data. The parameters are continuously updated with the arrival of newly collected data. Sometimes the formula needs to be modified or totally changed if the newly collected data reveal properties of the time series that were not found at the beginning. Here we introduce a simple layered prediction model to predict the future traffic. We can define as many layers as necessary. Here we choose three layers: day layer, hour layer and minor layer (minor layer can also be divided into layers like minute layer, second layer, millisecond layer, etc.). At each layer, we can use the existing model that suits our time series. All we need to do is to fill in the parameters according to our data. This layered prediction model is flexible. Each layer can have a different model, i.e., at different time scale the traffic can have different properties.

Day Layer:

In day layer, the day traffic is predicted by using the weekly traffic pattern and global trend. The model is expressed as $P = T + S + R$, where P is the predicted day traffic, T is the global trend, $S(i)$, i = *Sunday,…,Saturday*, is the determined seasonal part. The seasonal part has a period of a week. R is the random part within a range observing a distribution that is usually normal. The center of the range is zero.

By observing the traffic pattern in Figure 3.2.6, we find that there is a step change in Oct.12, 1999. The traffic before Oct.12, 1999 (low part) and after Oct.12, 1999 (high part) both show weekly periods, but have no increasing trend, so T = 0.

The statistical results of the seasonal part are shown in Table 3.5.1. The average weekly traffic pattern is shown in Figure 3.5.1, where 'w' means 'whole', which refers to the complete period from July 1, 1999 to Dec. 31, 1999.

Table 3.5.1 Statistical results of weekly traffic

1.0e+06 *		Sun.	Mon.	Tue.	Wed.	Thu.	Fri.	Sat.
Low Part	Mean (bytes)	1.17	2.11	2.56	3.12	2.98	2.90	1.50
	Error Range	(-0.34, 0.48)	(-1.14, 1.18)	(-1.91, 1.28)	(-0.42, 0.70)	(-0.96, 1.46)	(-0.76, 0.70)	(-0.40, 0.39)
	Var.	0.25	0.79	0.87	0.30	0.66	0.39	0.26
High Part	Mean (bytes)	3.13	5.81	5.75	6.35	6.69	6.22	3.73
	Var.	0.44	0.89	1.10	0.83	1.05	1.24	0.53
Whole Part	Mean (bytes)	2.07	3.81	4.05	4.73	4.76	4.49	2.52
	Var.	1.06	2.05	1.86	1.76	2.08	1.91	1.20

Christmas is in December, so the weekly pattern in the high part is more oscillated than the weekly pattern in the low part (see the variance in the table). And for that reason, we

only give the error ranges of the week pattern in low part. In the low part, there are 12 Wednesdays, the rest have 13 samples. It appears that the samples are not large enough, we find the errors are not normally distributed. To see the step change at Oct. 12, 1999 more clearly, we plot the average day traffic of each month in Figure 3.5.2. The patterns of Monday to Sunday are similar, we only show Monday traffic pattern in Figure 3.5.3.

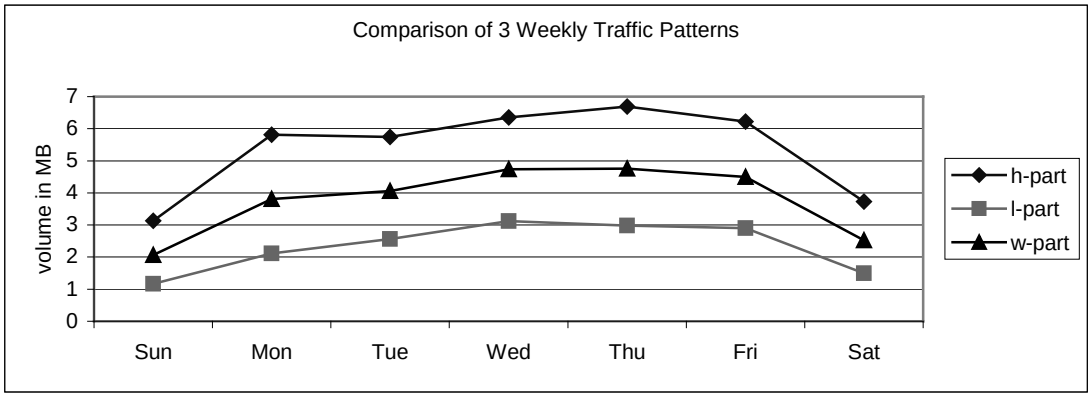

Figure 3.5.1

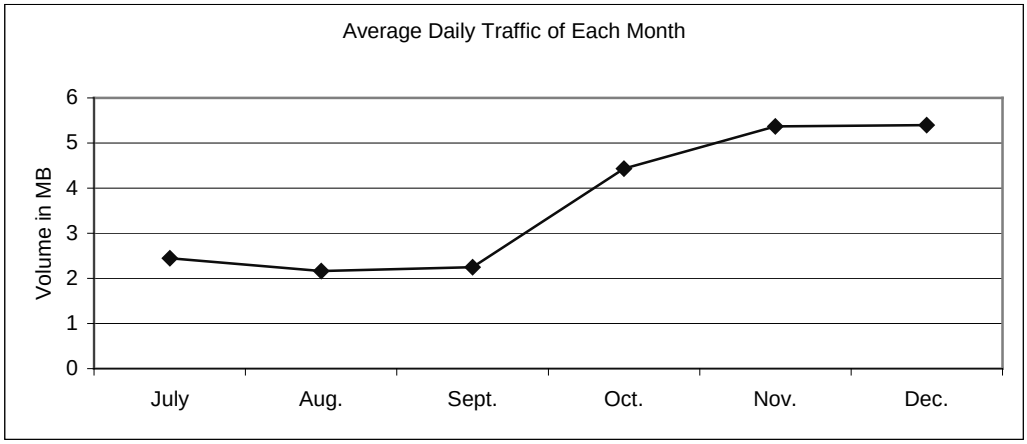

Figure 3.5.2

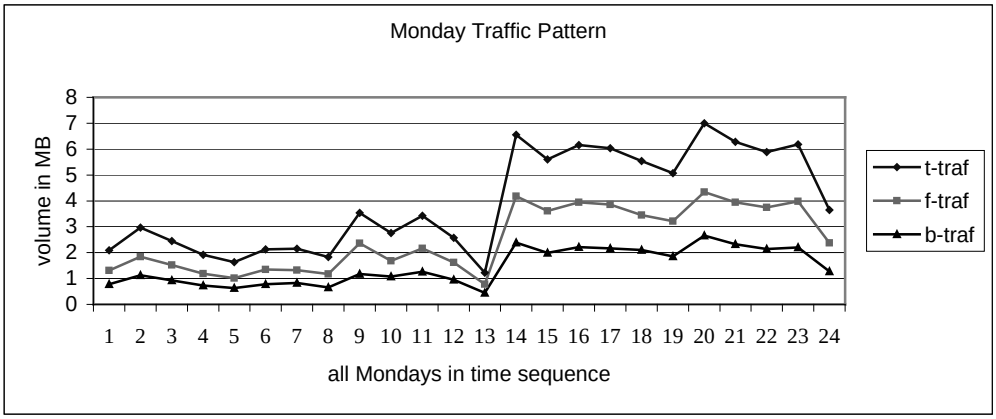

Figure 3.5.3

To predict the traffic of a day, we first determine which weekday it is. Assume it is Monday and it is in the low part period, then we can predict the traffic of the day by the value of Monday during the low part in Table 3.5.1, plus a random number within the range of -1.14 MB to 1.18 MB. The trend part T is 0 because there is no trend. As an example, we predict the hole that is missing in our trace files (see Table 3.5.2).

Table 3.5.2 Predicted daily traffic of the missing hole in the low part

	Seasonal Part	Random Part	Prediction
Aug. 25 (Wed.)	3.12 MB	0.23 MB	3.35 MB
Aug. 26 (Thurs.)	2.98 MB	-0.72 MB	2.26 MB
Aug. 27 (Fri.)	2.90 MB	-0.11 MB	2.79 MB
Aug. 28 (Sat.)	1.50 MB	0.31 MB	1.81 MB
Aug. 29 (Sun.)	1.17 MB	-0.01 MB	1.16 MB
Aug. 30 (Mon.)	2.11 MB	0.05 MB	2.16 MB
Aug. 31 (Tue.)	2.56 MB	0.15 MB	2.71 MB
Sept. 1 (Wed.)	3.12 MB	0.01 MB	3.13 MB
Sept. 2 (Thurs.)	2.98 MB	0.21 MB	3.19 MB
Sept. 3 (Fri.)	2.90 MB	-0.40 MB	2.50 MB
Sept. 4 (Sat.)	1.50 MB	-0.04 MB	1.46 MB
Sept. 5 (Sun.)	1.17 MB	0.44 MB	1.61 MB
Sept. 6 (Mon.)	2.11 MB	1.2 MB	3.31 MB

Figure 3.5.4 shows the predicted traffic. Figure 3.5.5 shows the traffic pattern after the hole is mended. We can see the hole is mended very well. According to the r-vlm (a

constant), we can get backward traffic and forward traffic. Similarly, we can predict the daily traffic after October 12, 1999 (high part).

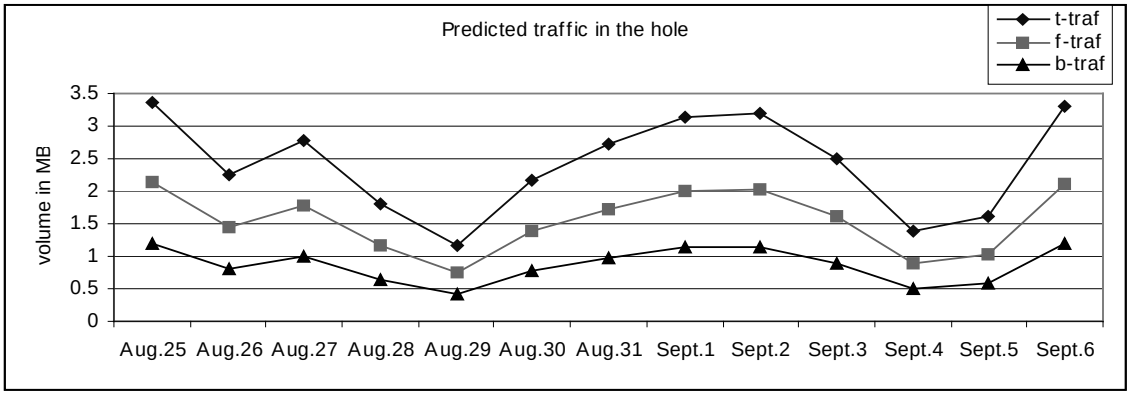

Figure 3.5.4

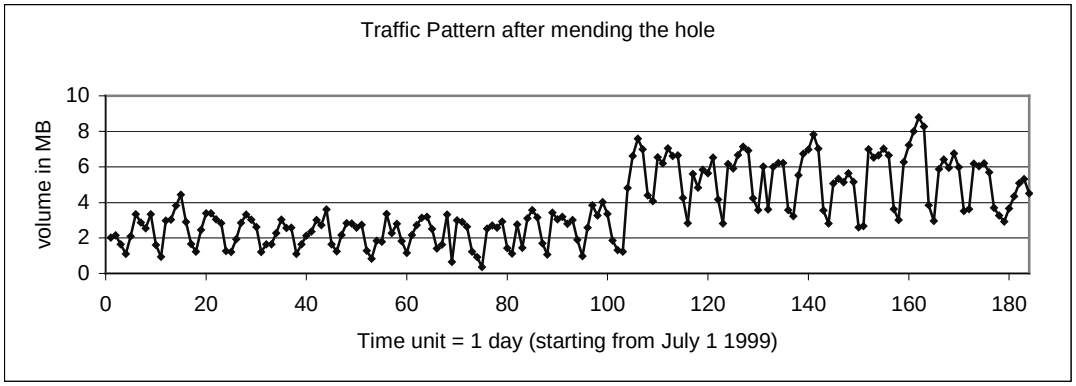

Figure 3.5.5

Hour Layer:

Suppose we obtained the predicted traffic of a day. The pattern of daily traffic is assumed to be the same for all days (an invariant). The ratio of forward traffic to backward traffic is a constant. Then we can obtain the traffic pattern of that day in hour time unit by splitting the total traffic among the different hours by their ratios. The ratio is provided in the following table.

Table3.5.3 Proportions of each hour traffic in daily traffic

Hour	Proportion
0 – 1	0.019
1 – 2	0.011
2 –3	0.0064
3 – 4	0.0044
4 – 5	0.0029
5 – 6	0.0035
6 – 7	0.0067
7 – 8	0.015
8 – 9	0.028
9 – 10	0.047
10 – 11	0.061
11 – 12	0.072
12 –13	0.074
13 – 14	0.075
14 – 15	0.078
15 – 16	0.079
16 – 17	0.078
17 – 18	0.069
18 – 19	0.060
19 – 20	0.057
20 – 21	0.045
21 – 22	0.042
22 – 23	0.037
23 – 24	0.029

As an example, the predicted volume of traffic for Sept. 6 is 3.31 MB. If we split 3.31 MB among the time intervals according to their proportions, we can get the daily traffic pattern. Also we can get backward traffic and forward traffic by using r-lm = 1.75, see Figure 3.5.6.

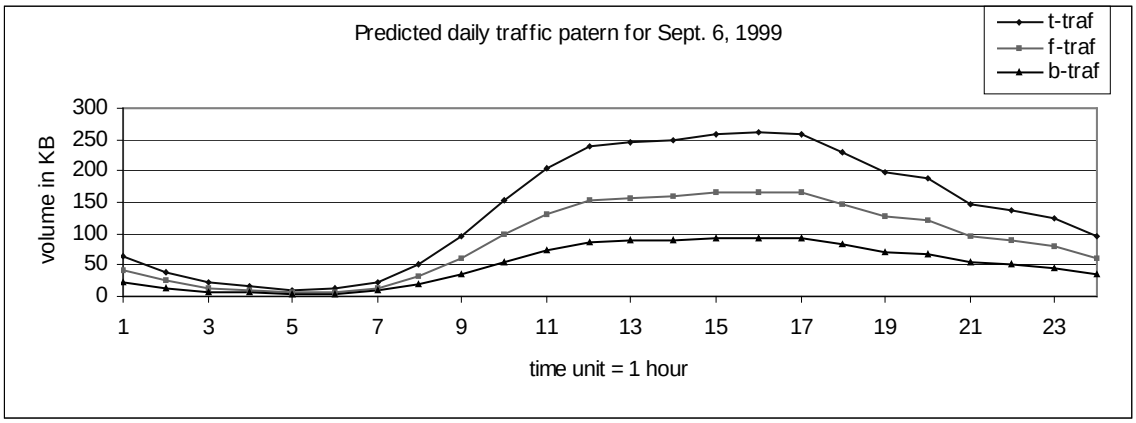

Figure 3.5.6

Minor Layer:

We can also predict the traffic in smaller time scales than 1 hour. Assume we know that traffic volume on a specific day from 10:00am to 11:00am is V bytes, and we want to know the flow pattern on a scale of a second. The former results guarantee us that traffic is self similar from 10:00am to 11:00am up to a scale of 15 minutes, so we can use an existing self-similar traffic producer to generate the traffic, with the condition that their aggregation is V bytes. We are not going to do this in this work, because we do not do traffic simulation here. In the following we give two examples of prediction at the day layer only.

Example 1: Forecast of the Number of Unique Client IP Addresses in January 2000

We already know that the number of different client IP addresses has a weekly period and a linear increase trend (after Sept. 7, 1999). We calculate the ratios for each day in a week for the period of Sept. 7, 1999 – Dec. 31, 1999 separately. The ratios are 11.6, 13.7, 12.1, 12.3, 10.0, 11.8 and 14.2 for Monday to Sunday respectively, with the dimension 1/week.

We use the average of all the ratios as the common ratio for each day in a week for simplicity. The common ratio is (11.6 + 13.7 + 12.1 + 12.3 + 10.0 + 11.8 + 14.2)/7 = 12.2 (of course we could also use these ratios separately). We use the model at the day layer (i.e., P = S + T + R). To reduce the error (influence of the Christmas in December), we use the average numbers of unique IP addresses of the last 3 weeks (from Dec. 10, 1999 to Dec. 31, 1999) as the determined weekly seasonal values. The weekly seasonal values are 357, 394, 407, 450, 464, 434, and 352 in the order of Sun., Mon., Tue., Wed., Thurs., Fri., and Sat. The trend part is linear, so T = 12.2 * j, where j indicates the j*th* week in Jan. 2000, j = 1,2,3,4,5. R is omitted for simplicity and its minor effect. For example, we predict the number of unique IP addresses on Jan. 28, 2000. Jan. 28, 2000 is the 4th Friday, so S = 434, j = 4, T = 12.2 * 4 = 48.8, and the predicted number of unique IP addresses on Jan. 28, 2000, is P = S + T = 482.8. Similarly, Jan. 29, 2000 is the 5th Saturday, so j = 5, and the predicted number of unique IP addresses on Jan. 29, 2000, is P = S + T = 352 + 12.2 * 5 = 413. The predicted numbers of unique client IP addresses and the actual data derived from our trace files for each day in Jan. 2000 is shown in Figure 3.5.7. The prediction is close to the real data.

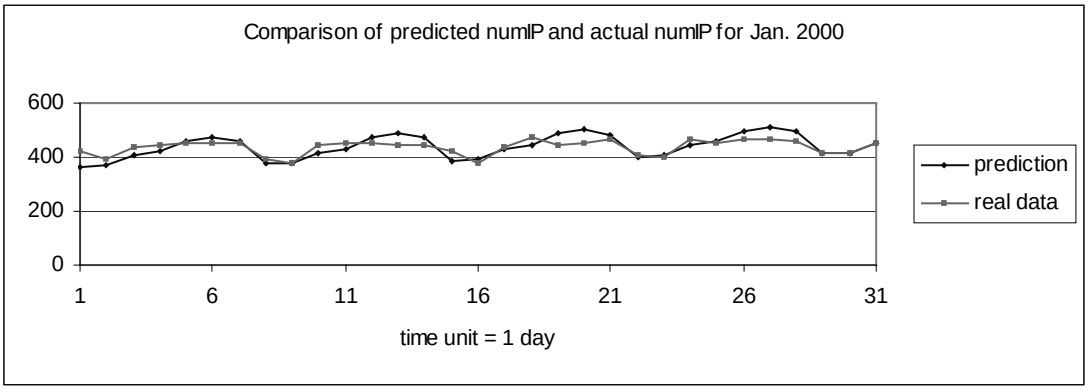

Figure 3.5.7

Example 2: Traffic Forecast for January 2000

The traffic increase trend of the high part is not obvious. The average daily traffic in December is 5.40e+6 bytes. The average daily traffic in November is 5.37e+6 bytes. Considering the effect of the Christmas in December, we use the average traffic of the first 23 days in December. It is 5.90e+6 bytes, lager than the former one as expected. We calculate the traffic increase ratio as (5.9 – 5.37)/4 = 0.133, with dimension of MB/week. We use 4 as the divisor because there are roughly 4 weeks in a month. We use the high part weekly traffic in Table 3.5.1 as the seasonal values. The trend part T = 0.133 * j, where j = 1,2,3,4,5. P = S + T, R is omitted for simplicity and its minor effects. The predicted traffic in January 2000 is shown in Figure 3.5.8. The prediction is less than the real data for the last two weeks, especially for the last week. That means we need to update the parameters for further forecasting. Preliminary study of the trace data collected since Jan. 2000 shows that the traffic increased more than linearly during the first four months of the year.

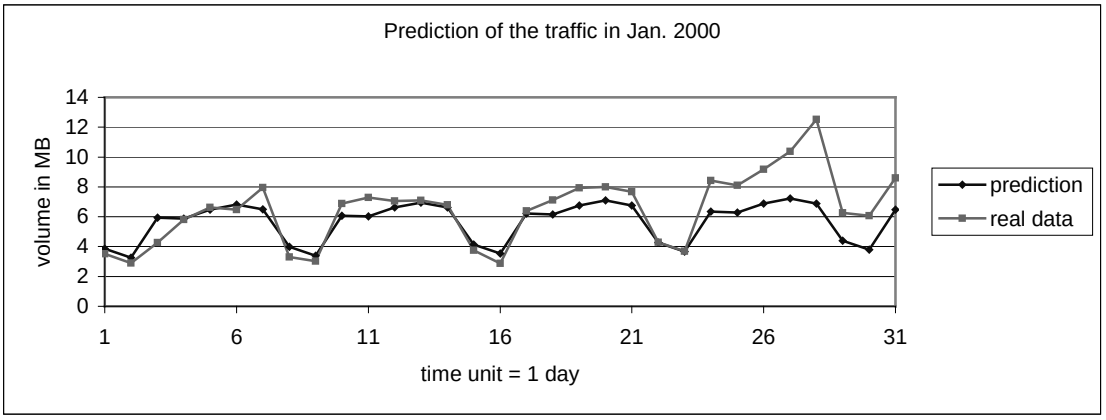

Figure 3.5.8

3.6 Summary

In this chapter we discussed properties of the traces that we collected:

- The number of unique IP addresses assigned to the client increases linearly with time after September 7, 1999. The slope is 1.8 with dimension 1/day. It also has a weekly period.
- Average packet sizes of each day are time-invariant.
- The ratio of forward traffic volume to backward traffic volume and the ratio of number of backward packets to number of forward packets are constant.
- Daily traffic pattern and weekly traffic pattern are obvious in both time and frequency domains. And the patterns do not change over time.
- The traffic is self-similar. H values range from 0.70 to 0.92.

We also provide a layered model to provide future predictions. The layered prediction model is flexible, easy to understand, and easy to use. The examples show that the model can provide reasonably exact predictions. They also indicate that prediction models by necessity are based on assumptions about long-term trends. The accuracy of predictions is limited by the accuracy of the long-term trend estimate.

Chapter 4: Session Analysis of Trace Files

In this chapter, we will do some further session analysis of trace files and discuss the results.

4.1 Necessary Terminology and Notations

First we give some definitions only valid in this work. This is necessary because the same term may have different meanings in different places. For example, ‘session’ often means a user’s complete application session in WWW traffic research. The session starts with the user starting a browser and ends with the close of the browser. In the middle of the session, the user can open several frames at the same time. But it needs much information about the users to seek the start and end of a user’s session from traces that are usually recorded on the user’s side. The trace files that we use in this work are measured on the server’s side. So we do not have information to distinguish different users. To catch the user behaviors as close as possible from our trace files, we give two definitions of a session (see the definitions below). A session should always be initiated by a request of a client, so we think the first definition of a session is more reasonable. Actually, these two definitions give almost the same results after comparison. Our trace files are the records of Mobile Browser applications in which the client IP address is assigned dynamically. We cannot distinguish different users by unique client IP address, but we use unique IP address to distinguish different Mobile Browser applications plus a timeout condition. Clearly it is not true all the times. For example, if the client takes a long time to read the contents on the Browser, or if the client does something else in between a Browser

application, the one complete application session will have more than one client IP address. But we believe these cases are not the dominant ones according to our experience in using browsers. The session we defined can represent a Mobile Browser application session reasonably well. Next we give the definitions of a session and some terminologies and notations that can help us describe our concepts clearly and concisely.

Forward link: a link that directs from the server to the client.

Backward link: a link that directs from the client to the server.

Critical interval: noted as *C-I*, is the one of the conditions in session definition. The time interval between any two consecutive packets in a session must be less than this value. In this work the critical value is set to be 90 seconds.

Session:

Definition 1: a session must satisfy the following three conditions: 1. It is composed of a series of packets that form the forward link traffic and the backward link traffic. The client IP addresses of packets must be the same one that implies they are from and to the same client. 2. The time interval between the any two consecutive packets must be less than 90 seconds, the critical interval (C-I). 3. The starting packet of the session must be the backward packet. This implies that a session always starts with the request of the client.

Definition 2: same as definition 1, except that the third condition is dropped.

Incomplete Session: an incomplete session is composed of a series of packets that are either forward packets or backward packets, but not both. The time interval between any two consecutive packets must be less than 90 seconds.

f-session: an incomplete session that contains only forward packets.

b-session: an incomplete session that contains only backward packets.

Hold time: noted as *h-t*. It indicates the time allocated to the link as full rate when there is no transmission in the link. In each session, the forward link and backward link are only dedicated to this session at full rate for the period of *h-t*, if there is no traffic. After *h-t*, the transmission capacities are used for other sessions and link bandwidth is reduced to 1/8 rate if there is still no traffic. *h-t* may be longer than 90 seconds.

Session start: noted as *s-s*. Let the load of the first packet in a session be *S* bytes, the time stamp of the packet is *t* minutes, and the transmission speed is *v* bytes/minute. $s\text{-}s = t - S/v$, if the first packet of the session is a backward packet. $s\text{-}s = t$, if the first packet is a forward packet.

Session end: noted as *s-e*. Let *t* be the time stamp of the last packet in a session, load of the packet is *S* bytes, and the transmission speed is *v* bytes/minute. $s\text{-}e = t$, if the last packet of the session is a backward packet. $s\text{-}s = t + S/v$, if the last packet is a forward packet. It means the session ends right after the arrival of the last packet, which can be caused by the power off. We do not consider the *h-t* after the last packet. Neglecting *h-t* after the last packet in a session allows us to capture how long the client is using the session.

Session size: noted as s-size, is the number of bytes contained in a session.

Session length: noted as *s-len*. We define $s\text{-}len = s\text{-}s - s\text{-}e$.

Forward session length: denoted as *f-len*, the full-rate period of the forward link in a session.

Backward session length: denoted as *b-len*, the full-rate period of the backward link in a session.

Backward link utilization: noted as *b-ut*, is the percentage of the time devoted to the backward link in full rate (or plus *1/8* rate period) during a session. See Figure 4.1.1 and the example.

Forward link utilization: noted as *f-ut*, is percentage of the time devoted to the forward link in full rate (or plus *1/8* rate period) during a session. See Figure 4.1.1 and the example.

Activity factor: either *b-fact* or *f-fact*.

b-fact: percentage of the *s-len* used in transmission of data in backward link.

f-fact: percentage of the *s-len* used in transmission of data in forward link.

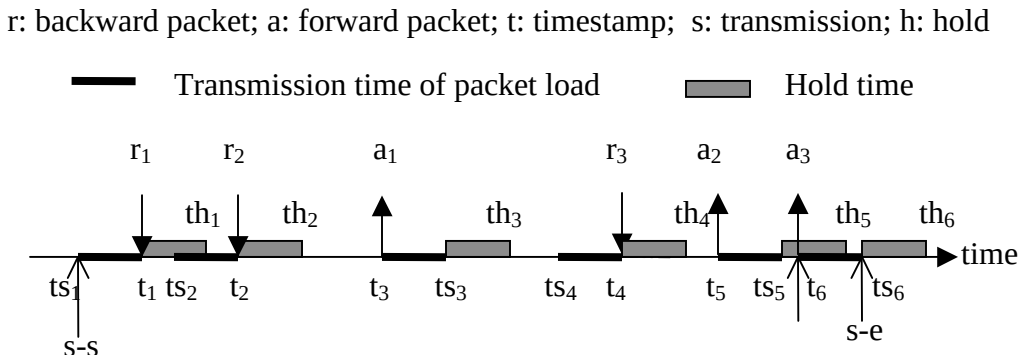

Figure 4.1.1

As an example, see Figure 4.1.1, where ‘t’ indicates timestamp, ‘th’ means the end of hold time, ‘ts’ means the start of transmission time for backward packets and the end of transmission time for forward packets. The figure shows a session, so the difference of

any two consecutive timestamps is less than 90 seconds. According to the definition above,

s- len = $ts_6 - ts_1$,

b-len = $(th_2 - ts_1) + (th_4 - ts_4)$, only full rate case;

f-len = $(th_3 - t_3) + (t_6 - t_5)$, only full rate case;

b-len = $(th_2 - ts_1) + (th_4 - ts_4) + (ts_4 - th_2)/8 + (ts_6 - th_4)/8$, full rate plus 1/8 rate case;

f-len = $(th_3 - t_3) + (t_6 - t_5) + (t_3 - ts_1)/8 + (t_5 - th_3)/8$, full rate plus 1/8 rate case;

b-ut = b-len/s-len,

f-ut = f-len/s-len,

b-fact = $((t_1 - ts_1) + (t_2 - ts_2) + (t_4 - ts_4))$/s-len,

f-fact = $((ts_3 - t_3) + (ts_5 - t_5) + (ts_6 - t_6))$/s-len.

4.2 Session Results

In this section we will do analysis of sessions and present the results. The sessions used in our analysis do not include the incomplete sessions, and when we say session, we mean a complete session that includes both backward and forward packets according to Definition 1, unless noted otherwise. We believe most of these incomplete sessions are aborted Mobile Browser applications. For example, the client may stop the browser for an unexpected interrupt or due to his/her impatience. First we show the soundness of neglecting the incomplete sessions. After investigating all the trace files from July 1, 1999 to December 31, 1999, we find an average day traffic volume is 3.8 MB, in which incomplete sessions account for only 0.4%, which is 12 KB. This implies that incomplete sessions can be neglected without influencing the analytical results much.

We are interested in various aspects in session analysis. Figure 4.2.1 shows the daily average number of sessions in each month. It is a step change pattern similar to the traffic pattern in Figure 3.5.2. We can conclude that the number of sessions per day is proportional to the traffic volume of that day. Figure 4.2.2 presents the average activity factor for each day, and clearly the end of December shows higher values than the rest. Contradictory to our initial intuition that would suggest that the average f-fact should be larger than the average b-fact (the fact that f-traf is larger than b-traf led us to this assumption), the result shows the reverse. This reveals the power and necessity of analysis, and proves that intuition cannot provide a sound ground for decision. We also find that the ratio of b-fact to f-fact is time-invariant, with a mean of 1.32 (see Figure 4.2.3). Figure 4.2.4 and Figure 4.2.5 show the link utilization for different values of h-t. The increasing trend is very obvious at the end of December, which is consistent with the higher activity values at the end of December. We can conclude that link utilization is proportional to activity factors. It is clear from the graphs that the link utilization increases with increasing lengths of h-t, and also oscillates less. We also calculate the utilization of links including periods at 1/8 rate, and the results are similar.

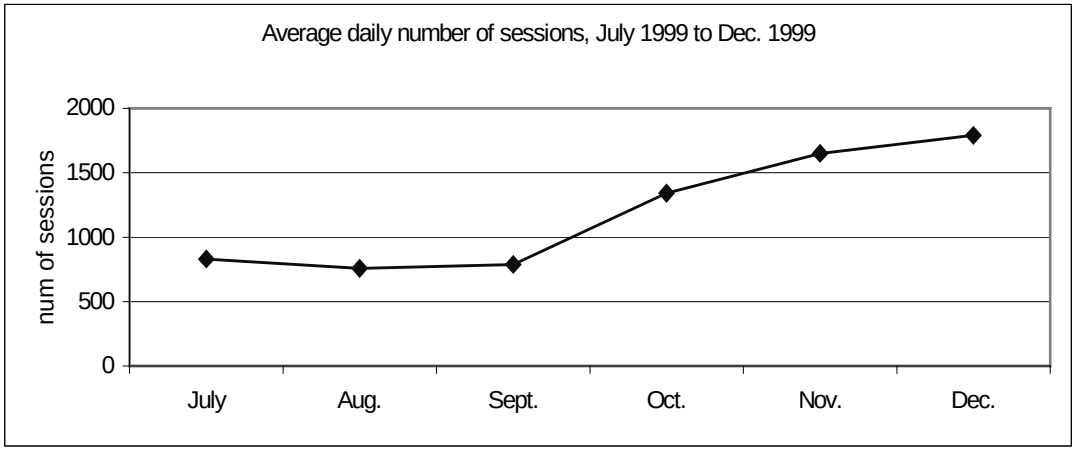

Figure 4.2.1

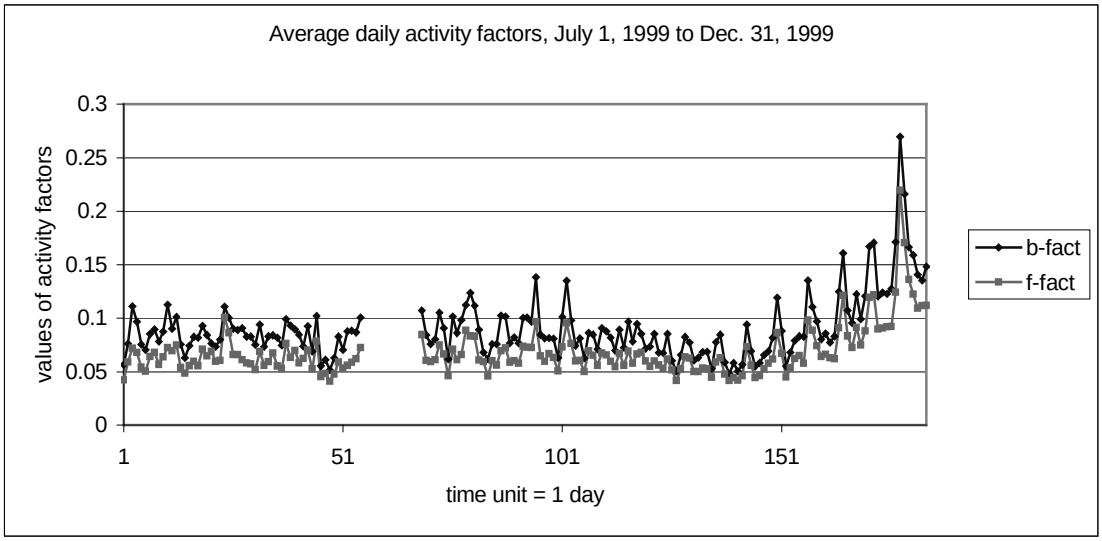

Figure 4.2.2

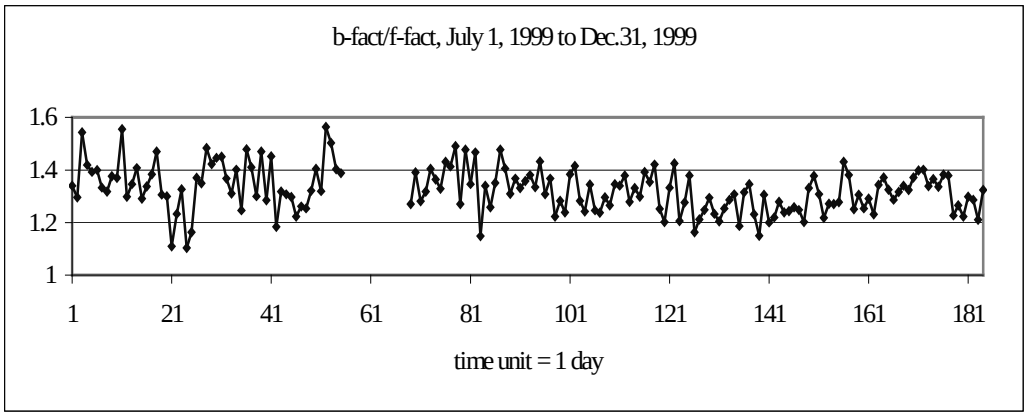

Figure 4.2.3

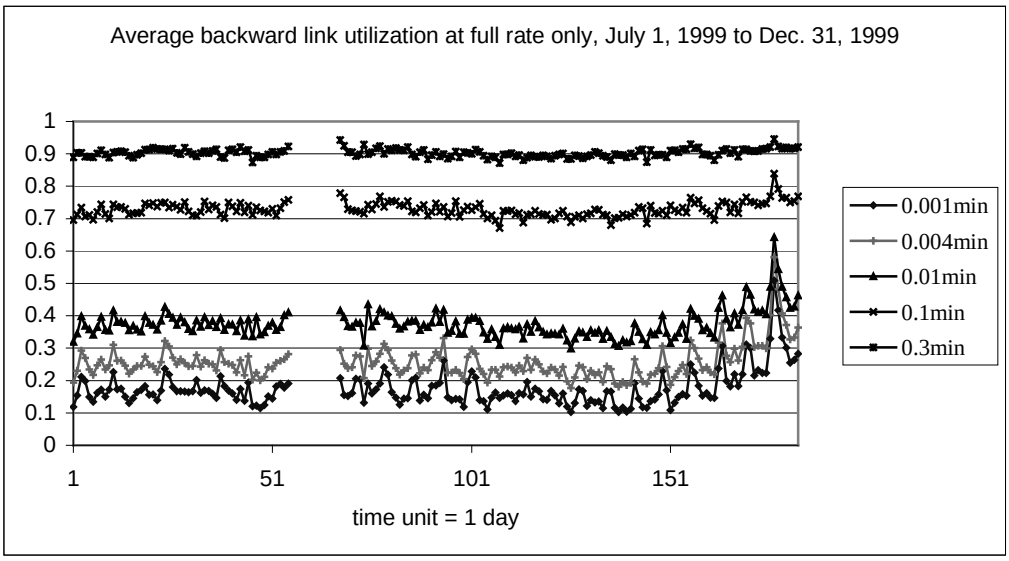

Figure 4.2.4

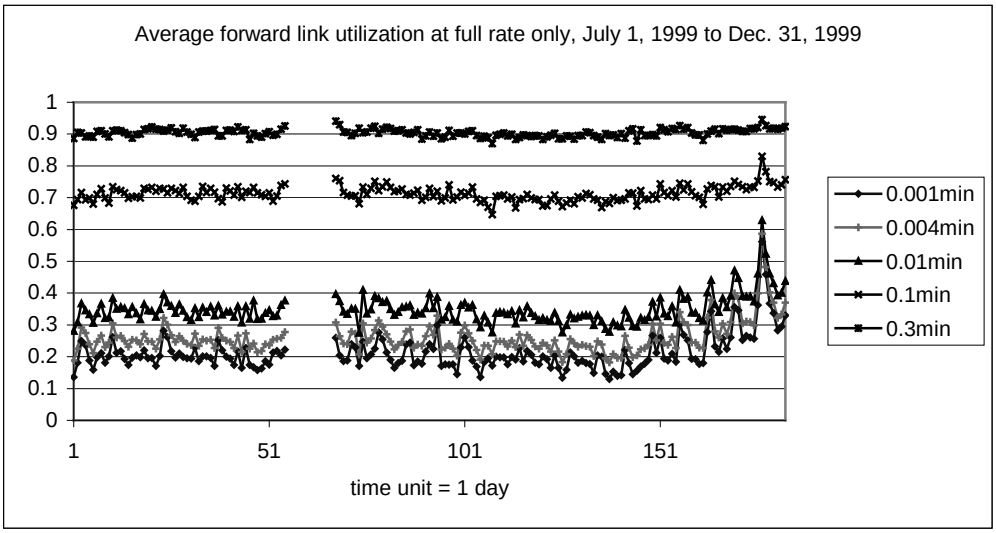

Figure 4.2.5

There are two important metrics of a session. One is session length, and the other is session size. After studying, we find that for each month around 80% of all sessions have a s-size that is less than 5000 bytes, around 90% of all sessions have a s-len of less than 4 minutes. And sessions with s-len that is less than 4 minutes account for about 65% of the traffic volume. This implies we cannot ignore 10% longer sessions because they account for 35% of the traffic volume. The average concurrent sessions during a day in each month have the same pattern as the daily traffic pattern. By concurrent sessions we mean that sessions occur at the same period and more strictly, they overlap in time. We only show the graphs for July 1999 to save space. The results in other months are similar (see Appendix A).

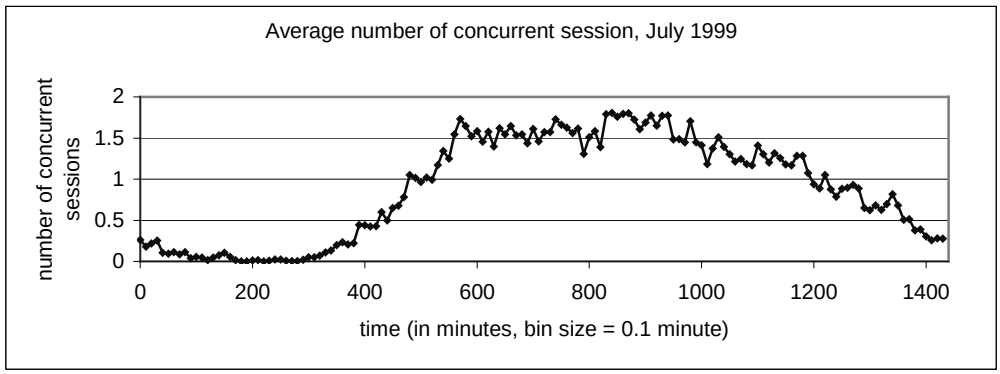

Figure 4.2.6

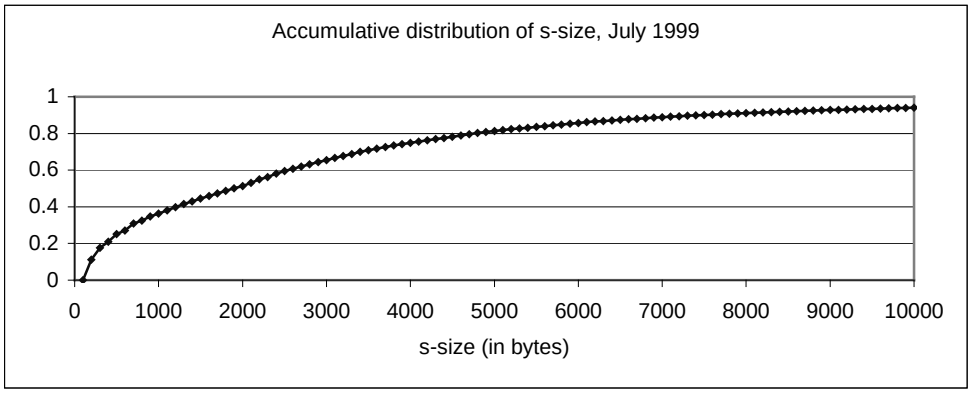

Figure 4.2.7

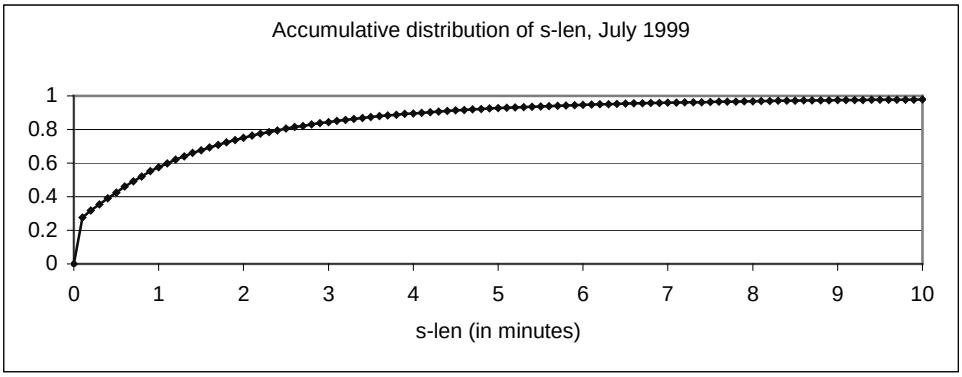

Figure 4.2.8

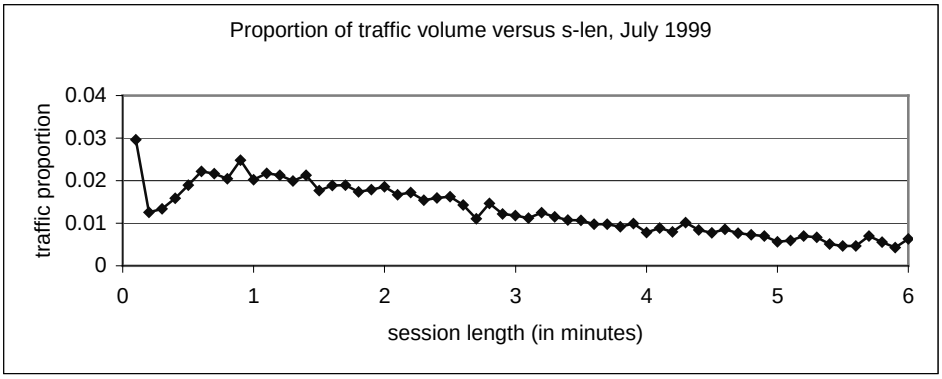

Figure 4.2.9

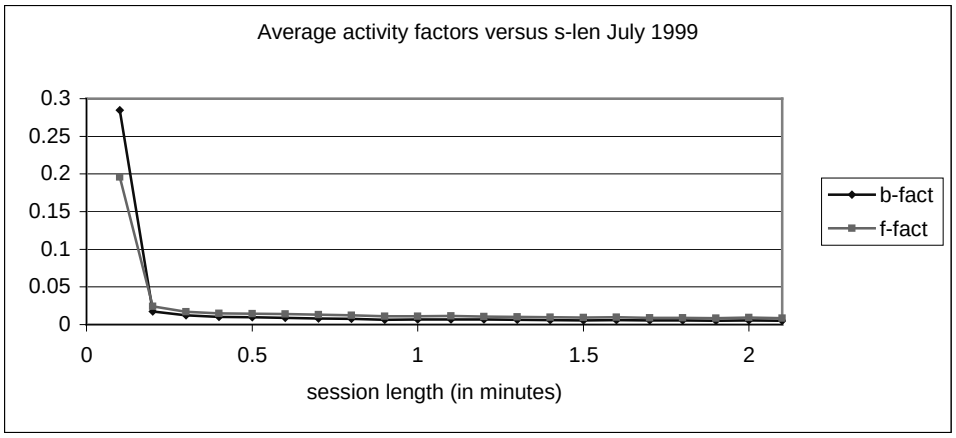

Figure 4.2.10

Figure 4.2.9 shows the average proportion of all the traffic volume that sessions with different s-len account for. We can see sessions with s-len <= 0.1 minutes account for the most part of the traffic volume of July 1999. The proportion decreases slowly after session length is greater than 1 minutes. The long sessions still account for quite a portion of traffic volume comparing to the short sessions. It is heavy tailed.

We like to say few more words about Figure 4.2.10 because it explains why the average b-fact is larger than the average f-fact. We notice that b-fact is higher than f-fact in shorter sessions, with s-len <= 0.1 minute, and we also know that a large percent of

sessions are short. For longer sessions, which are much fewer than short sessions, f-fact is larger than b-fact. The average, however, is dominated by the shorter sessions.

We also do session analysis based on Definition 2 of a session, and find the results almost do not change. In this work, critical interval (C-I) that is the second condition in session definition is set to be 90 seconds (90 seconds was a number given to us by Bell and also is based on timeout value in Microbrowser). We want to investigate the relationship between C-I and the number of sessions. It is clear that the number of sessions will decrease with the increase of C-I. We do this investigation for 24 trace files. Each month we randomly pick up 4 trace files, but make sure at least one is the trace file on a weekend. We believe this will give a conclusion that is valid for all trace files but saves us much laborious work. To save space, we just give test result on trace file of November 1, 1999. The result is shown in Figure 4.2.11. After 700 minutes, the number of sessions is almost constant, i.e., reaches the lower boundary that is the number of unique client IP addresses.

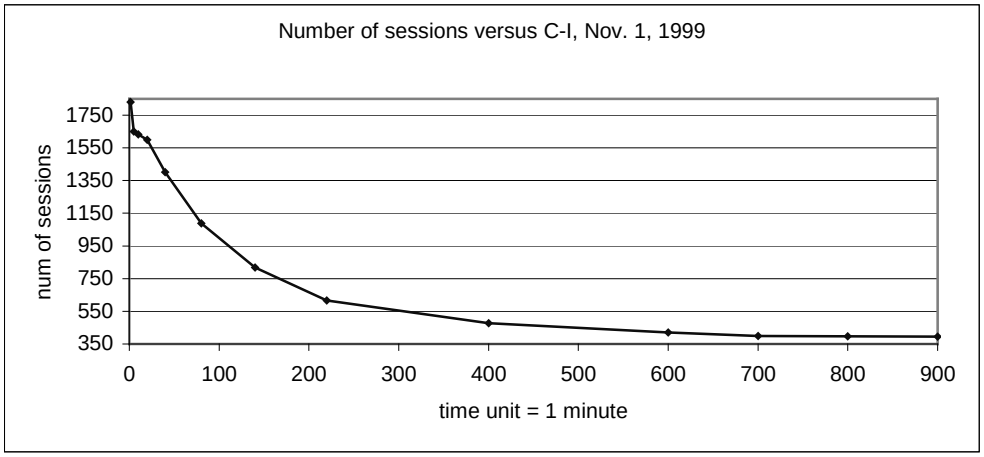

Figure 4.2.11

Some session results are summarized in Table 4.2.1.

Table 4.2.1 Session Analysis

	s-len < 4 minutes	Traffic of s-len<4m	s-size < 5 KB	MCSN	Max s-len (minutes)	Max s-size
July	90.0%	64.2%	81.6%	1.8	68, 0.13 MB	0.13 KB, 68
Aug.	89.3%	64.9%	82.0%	1.6	44, 83 KB	83 KB, 44
Sept.	89.6%	66.7%	81.2%	2.0	39, 72 KB	72 KB, 39
Oct.	87.3%	56.6%	79.0%	3.5	100, 0.18 MB	0.84 MB, 82
Nov.	88.3%	59.3%	80.1%	4.2	57, 64 KB	0.12 MB, 48
Dec.	90.0%	61.3%	81.2%	4.0	62, 0.17 MB	0.17 MB, 62

Remarks: The first column of the table is the percentage of number of sessions with s-len less than 4 minutes. The second column of the table is the percentage of the total traffic volume that the sessions with s-len less than 4 minutes account for. The third column of the table is the percentage of sessions with s-size < 5 KB. The fourth column of the table is the average maximum number of concurrent sessions in each month. The fifth column of the table is a pair of numbers in each cell. The first number is the maximum s-len. The second number is the s-size of the session with that s-len. The sixth column is similar to column 5, except the first number is the maximum s-size, and the second number is the s-len of the sesion with that s-size.

From Table 4.2.1, we can find that the maximum numbers of concurrent sessions in Oct., Nov., Dec., are larger than the ones in July, Aug. and Sept. We also know that the traffic in Oct., Nov., Dec., is higher than the one in July, Aug. and Sept. We can conclude, based on our data, that the maximum number of concurrent sessions of a day is

proportional to the traffic volume of that day. We also find that the session with maximum s-len is not always the session with maximum s-size (see cross of rows Oct. and Nov. and columns 5 and 6 in Table 4.2.1).

In the last part of this section, we show the average session length and session size of each day from July 1, 1999 to Dec. 31, 1999 in Figure 4.2.12 and Figure 4.2.13. The average session length from July 1, 1999 to Dec. 31, 1999 is 1.51 minutes. The average session size from July 1, 1999 to Dec. 31, 1999 is 3108.6 bytes. The average length of b-session is 0.391 minutes. The average size of b-session is 429.41 bytes. The average length of f-session is 0.275 minutes. The average size of b-session is 434.69 bytes. The total number of sessions is 209425. The total number of b-sessions is 2641. The total number of sessions is 1134. Again, we can see that b-sessions and f-sessions can be neglected, compared to sessions, in number and size.

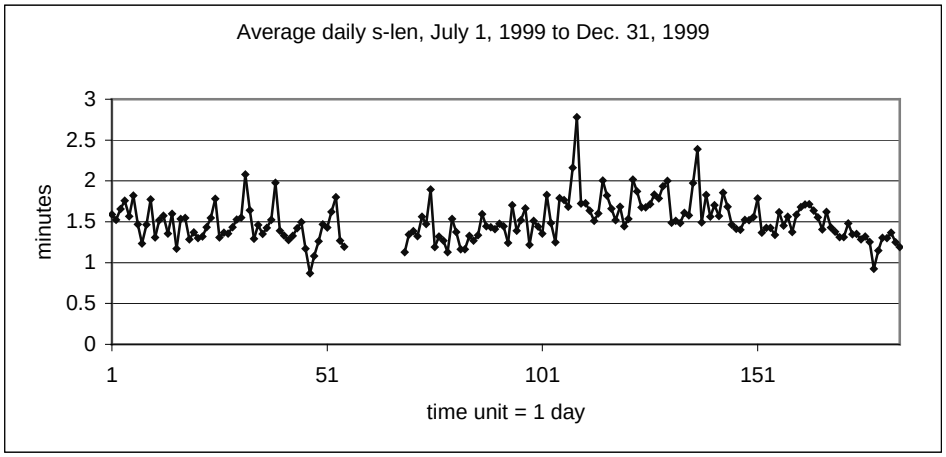

Figure 4.2.12

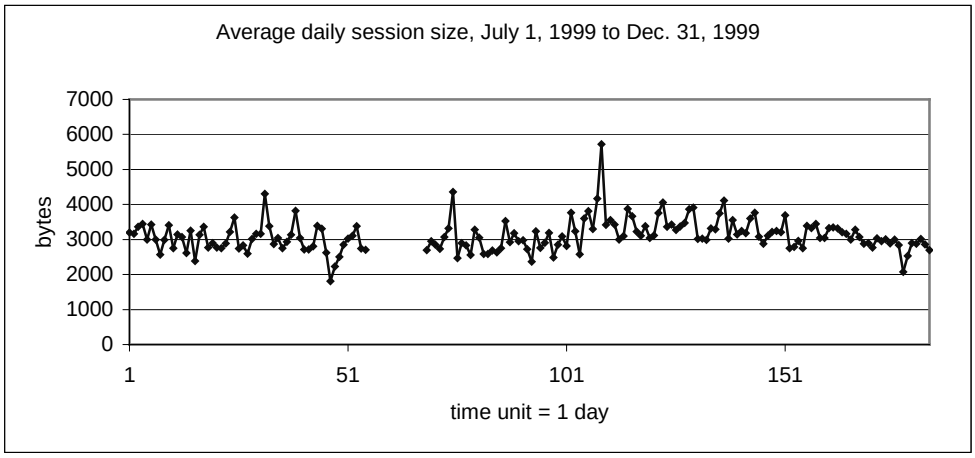

Figure 4.2.13

From Figure 4.2.12 and Figure 4.2.13, we can see that the average session length and the session size of each day have neither an increasing nor a decreasing trend. We can claim they are time-invariant.

We already know that the longest session does not necessarily have the largest size. But it is still interesting to see the relationship between the session length and the session size. We take 1 minute as the bin size, and let session length $L = n * 1$ minutes. We average the session sizes with session length $(n-1) < L <= n$ in bin L. Figure 4.2.14 shows the relationship. Session size linearly increases with session length until session length is 10 minutes. When session length is 10 minutes or longer, the session size increases less for a short period, then starts to decrease. This explains that the session size is not the only factor that will determine the length of a session. For example, slower users can cause longer sessions without generating much traffic.

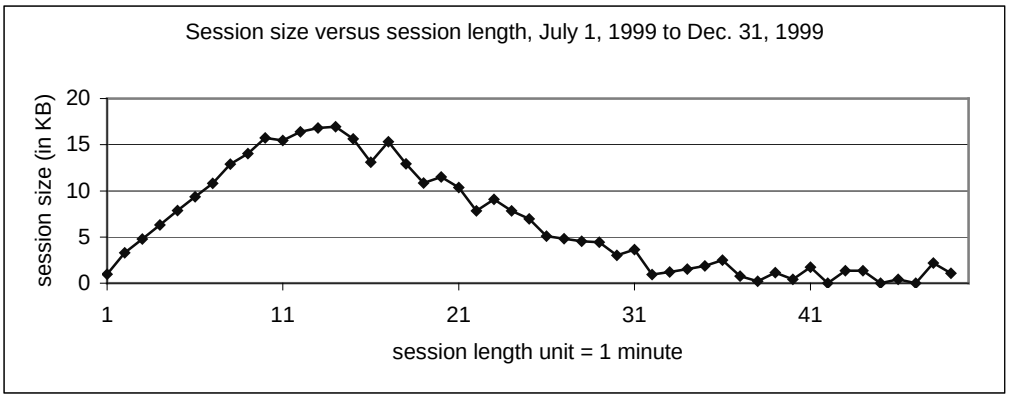

Figure 4.2.14

4.3 Parameters of a Session

This section will provide some useful parameters and information for Chapter 6 in which we will do performance studies. In order to solve a Layered Queuing Model for a WAP system, we need to provide necessary parameters. To make the model close to the practical case, we try to get as much useful information as possible from our trace files. We first introduce some parameters that we try to obtain from the trace files. Figure 4.3.1 shows a session composed of a series of backward and forward packets that imply the inquiries from the client and responses from the server. We define an inquiry (or a request) of the client as a series of backward packets in the order of timestamp, with the last backward packet followed by a forward packet, or no packets after it at all. Similarly, we define a response (or an answer) from the server, as a series of forward packets in the order of timestamp, with the last forward packet followed by a backward packet or no packets after it at all. To demonstrate these definitions, consider Figure 4.3.1. ‘r’ denotes

request packets and ‘a’ identifies answer packets. The first inquiry is composed of r11 and r12; the second request consists of only r21; the third request combines r31, r32 and r33. The first response is composed of a11 only; the second response is composed of a21 and a22.

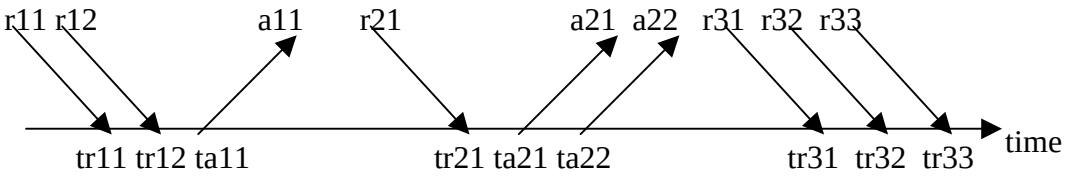

Figure 4.3.1

Some other parameters are defined in the following:

- Service time: the interval between the time that the server receives the request from the client and the time that the server sends out the response to the client.
- Synchronous request: the request of the client is answered with a response from the server.
- Asynchronous request: the request of the client is not followed by an answer from the server.
- Idle time: the interval between an answer from the server and the next request of the client.
- Request time: the interval between the first backward packet and the last backward packet in one request.
- Answer time: the interval between the first forward packet and the last forward packet in one response.

As an example, we give their expressions according to Figure 4.3.1.

Service time = ((ta11 – tr22) + (ta21 – tr21))/2

Idle time = ((tr21 – ta11) + (tr31 – ta22))/2

Number of synchronous requests is two

Number of asynchronous requests is one.

Request time = ((tr12 – tr11) + (tr21 – tr21) + (tr33 – tr31))/3

Answer time = ((ta11 - ta11) + (ta22 – ta21))/2

Based on the trace files from July 1, 1999 to Dec. 31, 1999, we obtain service time = 0.211 seconds, idle time = 0.856 seconds, request time = 3.881 seconds, answer time = 0.592 seconds, 11.19 synchronous requests and 0.766 asynchronous requests per session on average. We also find that the average request size is 87.31 bytes, the average response size is 152.3 bytes. From July 1, 1999 to Dec. 31, 1999, the average response size of each day, the average request size of each day, service time and idle time and synchronous and asynchronous request numbers of each day are all time invariant. The ratio of response size to request size is 1.744, and is very close to the ratio of forward traffic volume to backward traffic volume. We can find that the ratio of answer (response) size to request size increases with the session length, but does not change much with the session size.

We also find that the request size and response (answer) size increase with the session size but not with the session length. The number of synchronous requests linearly increases with the session size. When the session length is in the range of 1 minute to 10 minutes, the number of synchronous requests linearly increases with the session length. But when session length is beyond 10 minutes, the number of synchronous requests no longer increases with the session length and even decreases sharply when the session length is longer than 17 minutes, like the relationship between session size and session

length. This implies that WAP applications longer than 10 minutes may be caused by the habit of the user like slow reading or typing. Some additional relationships are revealed in the following figures.

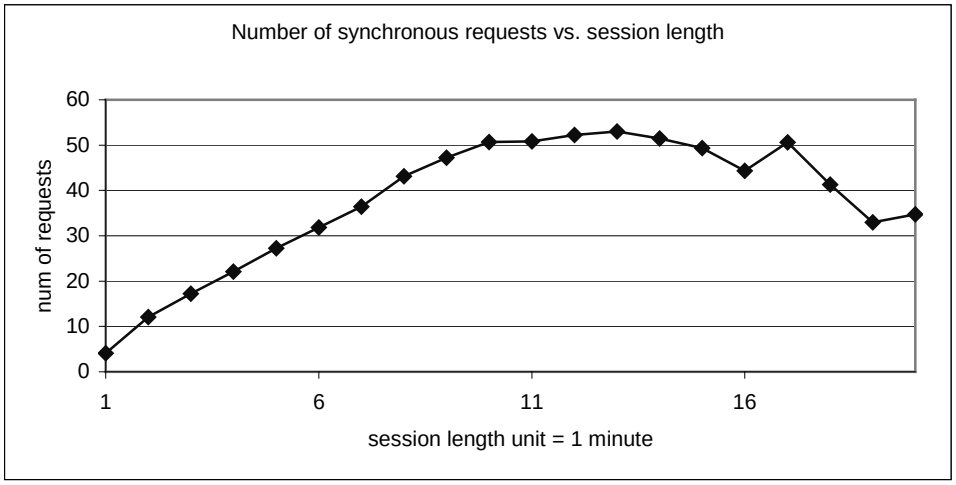

Figure 4.3.2

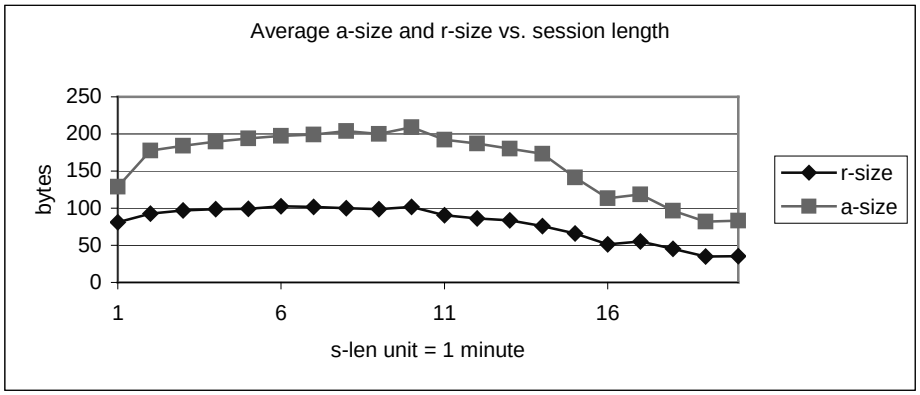

Figure 4.3.3

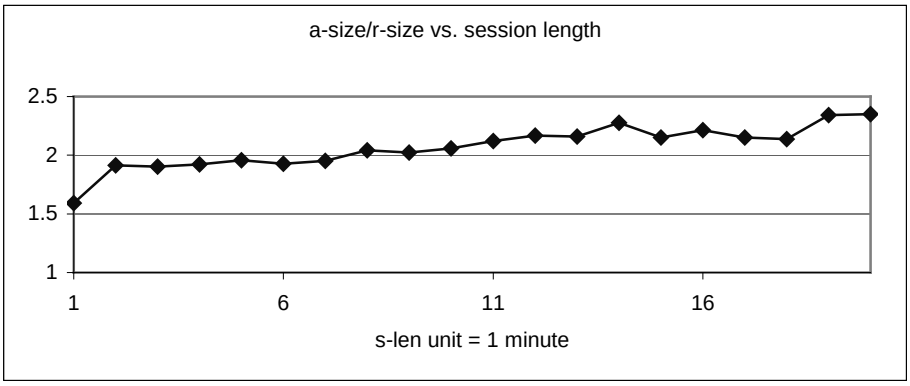

Figure 4.3.4

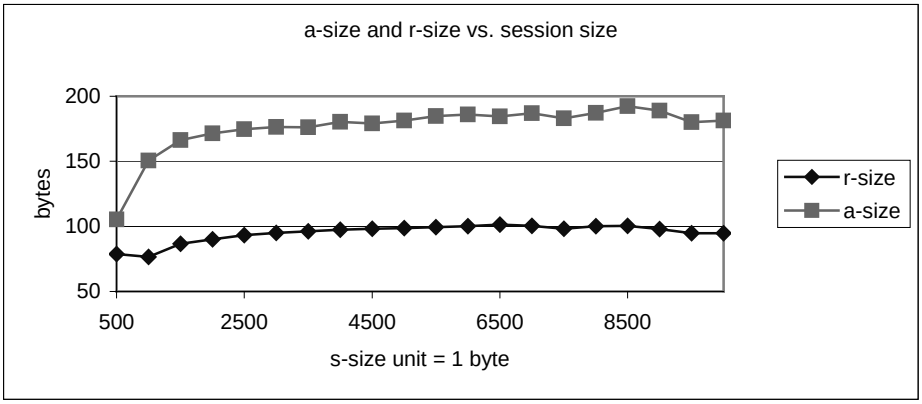

Figure 4.3.5

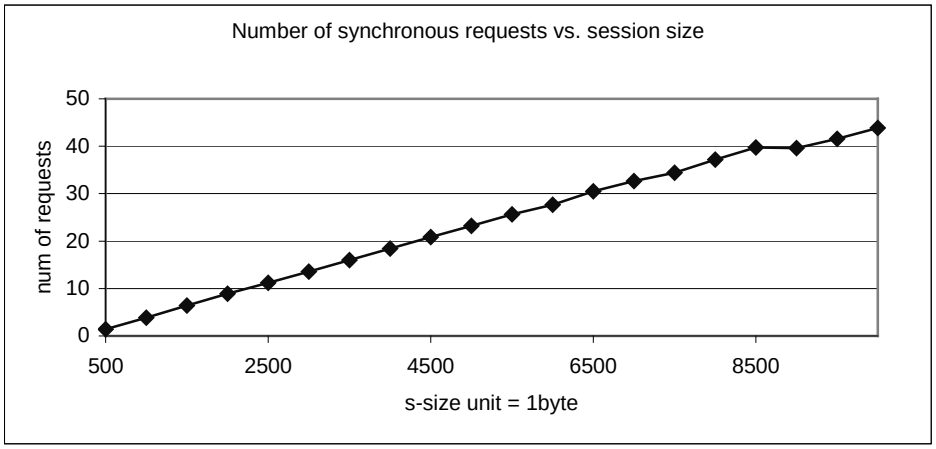

Figure 4.3.6

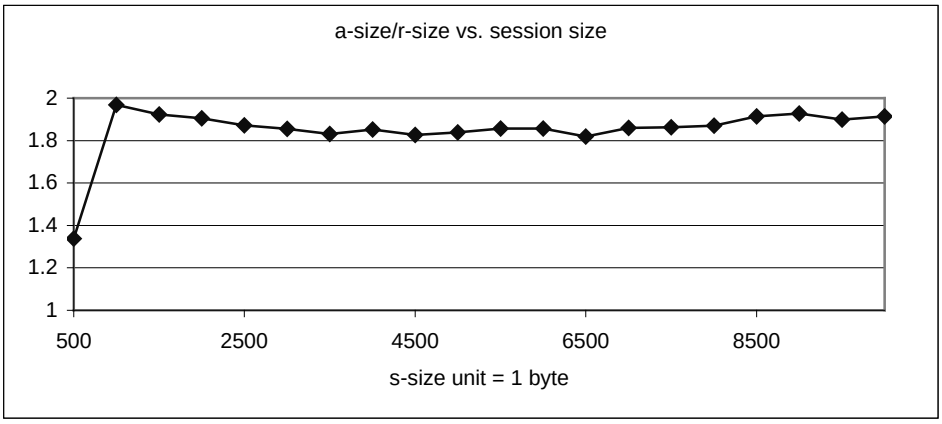

Figure 4.3.7

Parameters like the number of synchronous requests, the number of asynchronous requests, service time, idle time, request time and answer time, can be used in the performance models in Chapter 6. Relationship between the number of synchronous requests and the session length can be the basis for deducing the number of synchronous requests according to the length of a session. Relationship between the number of synchronous requests and the session size can be the basis for deducing the number of synchronous requests according to the size of a session. This gives us an option to define a type service provided by the gateway server as a session by the session length or by the session size. The relationships between request size, answer size, the ratio of the two, the session length and the session size will provide guidelines in the simulation of sessions.

4.4 Summary

In this chapter, we have done session analysis, and find:

- Two definitions of session give almost the same results.
- The traffic volume of incomplete sessions accounts for only 0.4% of total daily traffic volume, so incomplete sessions can be neglected.

- The activity factors, link utilization with full rate, and link utilization with full rate and 1/8 rate have similar patterns. There is an obvious increase at the end of December 1999.
- The ratio of b-fact to f-fact, the ratio of request size to the answer size, the average session size and average session length of each day are time invariant.
- The number of sessions increases since September 1999. This might be caused by the increase of user population and more frequent use of the system.
- The number of sessions of a day and the maximum number of concurrent sessions of a day is proportional to the traffic volume of that day. So we deduce that the number of sessions of each day should also have a weekly period. By spectral analysis, its weekly period is revealed without any doubt.
- 90% of sessions are less than 4 minutes. This implies that WAP users are more intent to use short WAP applications. This is coincident with what we know about the WAP users. They usually use WAP to check and send emails, check new information about sports and stock prices, etc. They seldom use WAP to download large files or read novels.
- Even though the longer sessions are few in numbers, they account for a big portion of traffic.
- The number of sessions decreases with the lengthening of C-I.
- If we divide the session length into 0.1 minutes interval, the largest proportion of sessions has session length within 0.1 minutes.
- If we divide the session size into 100 bytes interval, sessions with 100 bytes < s-size <200 bytes are the most.

- The session length is not the only factor that will determine the session size. After the session length exceeds 14 minutes, the session size starts to decrease.
- The relationship between the number of synchronous requests and the session length is similar to the relationship between the session size and the session length.
- The number of synchronous requests increases linearly with the session size.

We calculated the average session sizes of each month. From July to December they are 2.95 KB, 2.86KB, 2.86 KB, 3.31 KB, 3.25 KB, 3.02 KB. The values are around 3. But we need more data to claim it as a constant.

We also calculated the average daily traffic of each month, noted as *d-traf*, and the average daily maximum number of concurrent sessions of each month, noted as *m-ses*. From July to December *d-traf/m-ses* is 1.362 MB, 1.353 MB, 1.126 MB, 1.266 MB, 1.278 MB and 1.35 MB respectively. These values are around 1.3. But we need more data to claim that *d-traf/m-ses* of each month is time-invariant. If *d-traf/m-ses* were a constant, we could deduce the maximum number of concurrent sessions of a day by the traffic volume of that day.

Chapter 5: Comparison of the Results

In this chapter we will compare our results with related results reported in the literature. A lot of work has been done analyzing all kinds of traces. We only choose those that relate to our work closely.

5.1 Traffic Pattern

[Zhu 1994] presented the results from measurements of wide-area network TCP conversations between the Campus Ethernet at the University of Saskatchewan and the Internet in 1994. The author used traces of a four-day period, which contained 103,016 TCP conversations. The author showed that SMTP, FTP-Data, WWW, TELNET, etc., accounted for the majority of the conversations and the majority of the bytes transferred. The figures of number of conversations vs. time during a day for SMTP, TELNET and FTP show a similar pattern to the daily traffic pattern in our work. [Paxson 1995] studied 24 wide-area traces by investigating a number of wide-area TCP arrival processes such as FTP data connection arrivals within FTP sessions and TELNET packet arrivals. In the paper, for various protocols, the authors plot for each hour the fraction of an entire day's connections of that protocol occurring during that hour. For TELNET and SMTP, the pattern is very similar to the daily traffic pattern in our work. But their pattern also revealed the lunch-related dip at noontime, which our daily traffic pattern does not show. Though longer connection time does not necessarily mean that heavier traffic will be produced, generally it does produce more traffic. We can deduce that the daily traffic pattern of TELNET and SMTP is similar to our daily traffic pattern. [Thompson 1997] measured traffic on the OC3 link of the U.S. East Coast network. For the domestic link

(traffic within the U.S.), during a day the traffic is low in the early morning when most users are in bed. The traffic increases from morning to noon. Then the traffic keeps stable for few hours and starts to decrease till early next morning. During a week, the traffic is low on weekends when most users enjoy outdoor activities, and the traffic is high on weekdays. The daily pattern and weekly pattern that were revealed by the authors are very similar to the ones in our work. The range of traffic flow they measured is from 10 Mega bits/sec to 55 Mega bits/sec, which is much bigger than the traffic flow studied in this work. This is because the OC3 link has much wider bandwidth than wireless links. The traffic patterns on the external link (traffic to and from other countries than U.S.) are different from the ones on the domestic link. The cause may be that the different time zones in different areas and countries make the summation of the same user behavior in different areas and countries different from the user's behavior in one area that has one time zone or several time zones that are close. For example, one area's early morning might be another country's afternoon. The summation of the two traffic patterns will not reflect the fact that the traffic in the early morning is light. That paper also described many components of the traffic, like WWW, DNS, SMTP, FTP, TELNET, etc, which cannot be distinguished in our traces. [Arlitt 1999b] studied the workload on the Internet Service Provider (ISP) that provides interactive data services to residential and business subscribers. They collected the traces of the proxy server from Jan.3, 1997 to May 31, 1997 on a daily basis. During the data collection period several thousand users used the system. This paper presented a different daily traffic pattern. During a day, the traffic volume level reaches the bottom in the early morning (the same as in our results), and then the traffic increases till evening. The peak load appears in the evening, which is

different from our result and the result in [Thompson 1997]. The weekly pattern they revealed is different from ours as well. They reported that traffic volume is heavier on weekends than the one on weekdays. The difference is caused by the different behavior of users using the services. [Arlitt 1999b] reported that most subscribers like to use the services when they are at home. That explains the peak load in the evening when people are often at home and not in bed yet and the peak load on weekends when most subscribers do not work and are likely at home. So we can see the user behavior is an important factor that influence the traffic pattern and the importance of studying the users' behaviors.

[Abdulla 1997a] studied the traffic patterns from both time and frequency domains. Their traces were collected from a Korean proxy server, a school library, CS, and the Engineering building. The daily and weekly periods are easily observed in the time domain. Through spectral analysis, they confirmed the daily and weekly periods and also found half-day and half-week periods. They did not reveal a monthly period because of shortness of the traces. To do spectral analysis of traffic, we choose traces from November 1, 1999 to December 31,1999, because the trace files are continuous in these two months and the traffic in these two months is stable. The results show half-day, day, half-week, week and month periods. If the sampling time unit is hour, the day period is the main component. If the sampling time unit is day, the week period is the main component. The day and week traffic patterns are similar to the results in [Abdulla 1997a]. [Abdulla 1997a] used Fourier series to express the seasonal part (day and week periods) that needs more complex mathematical operations, and then subtract the seasonal part and the mean part from the total time series. The residual is modeled with

the Weibull distribution. The residual is similar to the random part in our work, which is modeled with a normal distribution. The random part has only a minor effect in our work. In our work we characterize the traffic of each day in a week separately, in this way the weekly-period time series is decomposed into seven non-weekly-period time series. And the day period is expressed with a sequence of 24 discrete numbers, each of which is the traffic in an hour interval of a day. The daily traffic pattern is obtained by averaging all day traffic over the period from July 1, 1999 to Dec. 31, 1999. The method that is provided in this work is simple and flexible.

[Leland 1994] studied the traces collected on several Ethernet LAN's at the Bellcore Morristown Research and Engineering Center from August 1989 to February 1992. They reported self-similarity of the traffic and gave $H = 0.8$, after analyzing a sample of 27 hour (sampling time interval is 10 ms) by variance method, R/S plot and periodogram plot. [Paxson 1995] also reported the self-similarity of the network traffic. They used 24 traces of wide-area TCP traffic at Bellcore, coNCert, UK-US, etc., but did not give the H value. [Crovella 1997a] revealed a range of H between 0.7 and 0.8 based on the analysis of busy hours. The trace data was collected at the Computer Science Department, Boston University in 1995. They modified Mosaic to record the URL of each file accessed by the Mosaic user. Timestamps were accurate to 10ms. They claimed that estimates of H seem to decline when moving from the busier hours to the less-busy hours. [Abdulla 1997b] considered self-similarity as an Invariant for Web Proxies, with a range of H from 0.59 to 0.94. The authors also provided some results for some tested workloads. The values for the Hurst parameter are from 0.59 to 0.94. [Arlitt 1996] discovered a value of $H = 0.65$ in the ClarkNet data set, very small ($H = 0.53$) in the

Saskatchewan data set, and none at all in the Waterloo data set. [Choi 1999] provided a traffic generator model. They showed that $H = 0.805$ for the trace, and 0.78 for the model produced trace, by variance time plots, $H = 0.8$ for the trace and 0.77 for the model produced trace by R/S plot. The trace was recorded on the backbone network of the Georgia Tech campus from 11 AM to 12 PM on Wednesday October 7 1998.

In our work, we use absolute value method and variance method to calculate H values of our traces. We did this for each daily trace file. For each day trace, we define a low traffic part and a high traffic part. To examine the effect of different time intervals may bring, we define low traffic as 00:00AM to 11:00AM and high traffic as 11:00AM to 10:00PM for the first time. We define low traffic as 1:00AM to 6:00AM and high traffic as 10:00AM to 3:00PM for the second time. We calculate H values for the time intervals separately and found no obvious difference. We found H values range from 0.70 to 0.92. The accumulated time scale is up to 20 minutes. We also calculate H values separately for the traffic series from July 1, 1999 to August 24, 1999 ($H = 0.84$), and the traffic series from September 14, 1999 to December 31, 1999 ($H = 0.86$). The accumulated time scale is up to one hour. The results imply that the WAP traffic studied in this work is self-similar.

5.2 Session Model

Based on the different extents of information in trace files, different session models are devised to study the traffic in order to catch the user's behaviors as close as possible. [Zhu 1994] and [Paxson 1995] used a complete TCP conversation to depict the users' behavior. A conversation is defined as the communication between an application

program on one Internet host and an application program on another Internet host. A TCP conversation is initiated with the 'three-way handshake'. The conversation ends with FIN packets. A lot of detailed information is required to identify a TCP conversation from traces. We do not have enough information in our traces. Our definition of a session can be used to compare to the conversation. [Zhu 1994] and [Paxson 1995] revealed that during a day the number of active conversations for SMTP and TELNET is the lowest from 1:00AM to 5:00AM, and start to increase till 11:00AM, keeps stable for few hours and starts to decrease at 2:00PM till midnight. The pattern is very similar to the pattern that describes the number of concurrent sessions during a day in our work. [Zhu 1994] also gave the ratio of forward traffic to backward traffic for various protocols. The ratio for SMTP is 0.125, much lower than 1.75, the ratio in our work. The ratio for WWW is 19 and the ratio for TELNET is 61, which are much higher than our result. The difference may be the result of the differences among WAP, WWW, SMTP and TELNET. [Zhu 1994] revealed that the average conversation size for WWW is 14 KB, which is bigger than the average session size 3.1KB in our work. This is because WWW has wider bandwidth than WAP, so WWW users are more likely to use applications that will produce heavy traffic. For example, people are much less likely to read novels on a PDA than on a desktop. The author also showed that the average duration of the conversation for WWW is 13 seconds, shorter than the average session length of 91 seconds in our work. The reason for the difference may be that the WWW has faster transmission speed than WAP.

[Cunhua 1995] defined a session as a single execution of Mosaic. The trace files they used are collected at the application level. The authors modified Mosaic to record

the Uniform Resource Locator of each file accessed by the Mosaic user, the time that the file was accessed and the time required to transfer the file from its server. The traces could distinguish whether a file came from the Mosaic cache or was downloaded in a file transfer. Mosaic now is an obsolete WWW browser. Netscape is the most popular WWW browser. Netscape is not available in source form, which makes recording traces at clients' side almost impossible, so it is no wonder that the number of references about the traces at the clients' side is limited. [Cunhua 1995] showed the histogram of the number of sessions per day from Nov 1994 to March 1995. The number keep flat from Nov 1994 to early Jan. 1995, then increased till middle of Feb. 1995, and finally started to decrease. The pattern may well reflect the birth-to-death process of Mosaic itself. The number of sessions in our work increases over time.

[Barford 1997] also used the same session definition as in [Cunhua 1995], but adopted a Web page as the basic unit of the response to the request from the client. To locate the boundary between Web pages, they modified Mosaic to record user information. The authors built a benchmark, SURGE based on this session model. Now since tools like Java-script enable a response of multiple pages to one user's request, their methods to determine a session from the traces are not accurate any more. The formula that a request will result in a single page to be transferred is no longer true.

[Choi 1999] presented a session model that includes the case that one request will result in a response of multiple Web pages. The model defines a new unit, a Web-request that consists of a Web page or a set of Web pages. Information in the HTTP header and the TCP header help to determine the boundary of a Web-request and other parameters they defined such as In-line Object or Temporal Locality. Their model also incorporated

the Web caching effect. Their results show that the model can mimic the user's behavior in real life very closely.

[Arlitt 1999a] defined a user session model based on the TCP connection mechanism. They use the unique IP address in the access log to identify a distinct client. In a session all requests are from a single client to the World Cup Web site, with the interval between two consecutive requests from the client that has a unique IP address less than a certain value that is called timeout value. This user session definition is very close to our session definition 1, except that it only considers requests from the clients. They did not include the timeout value in the session length either. Like our results, the authors revealed that the session length increases with the bigger timeout value that we call critical value (C-I) in this work, as expected. They also showed that the number of sessions decreases with the increase of the timeout value. The curve shapes are similar to ours. But the number of sessions in their work is much bigger than the one in our work. The reason is that the system studied in their paper has a much wider bandwidth than the system we studied in this work. In this work the trace files are collected at IP level. We do not have enough information to define a complex model with many parameters. We give two definitions of a session in Chapter 4, which imitates the process of a client making queries to the Gateway server through Mobile Browser. We also give the definitions of a request and a response in Chapter 6. [Arlitt 1999a] showed that the number of sessions increased as the World Cup tournament progressed, with three big numbers in the two semifinal and the final matches. It is common sense that people pay more attention to the semifinal and final than to those first primary leg matches. This

means that the known behaviors of users can be important factors to predict the traffic over the Internet.

5.3 Character Distributions

Due to the difference of the information available in the respective traces, our parameter "request" is not directly comparable to the Web-request mentioned above. Because of the limitation of the information in our trace files, we cannot provide parameters such as file sizes, file access frequency and number of sessions per user. But we can still derive something comparable. The distribution of session size is similar to the distribution of document size in [Cunha 1995], [Crovella 1995] and [Barford 1997]. They follow a power law distribution that takes a hyperbolic shape. It can be expressed as $f(x) \sim x^{-a}$. When a is between 0 and 2, the distribution is called heavy tailed. For the file size distribution [Cunhua 1995] gave $a = 1.35$, [Crovella 1995] presented $a = 1.0$, [Arlitt 1996] reported $0.4 < a < 0.63$, [Barford 1997] reported $a = 1.0$ for W95 data set, $a = 1.47$ for W98 data set. In our work, session size distribution gives $a = 1.1$. The distribution of inter-arrival time of backward packets follows a power distribution, heavy tailed with $a = 1.8$, like the distribution of ON times in [Crovella 1995] with $a = 1.21$. [Arlitt 1996] showed that there are very few files less than 100 bytes at Web sites, and 10% of the files are larger than 100,000 bytes, most files are in the range of 100 bytes – 100,000 bytes. Correspondingly, we have a very small number of sessions less than 200 bytes and 10% of sessions larger than 8,000 bytes. Like [Arlitt 1996] and [Abdulla 1997c], we also find some invariants (see Section 3.4) that are valid in this work.

5.4 Summary

In this chapter we compared our results to other related results. Our traces are from the WAP applications, and recorded at the IP packet level. The trace file contains limited information. Many WWW trace files are collected at HTTP proxies or Web servers that also miss information like session start, session end, or which document is viewed out of the browser's cache. To make more accurate session model, many traces at different levels and at different location (client, proxy and server) need to be recorded. [Choi 1999] included many trace files at different levels in their study.

We also find the general existence of daily and weekly periods in network traffic. But the difference in the behavior of user's using services makes the daily traffic pattern and weekly traffic pattern different. We can also see that WWW traffic and WAP traffic in this work display self-similarity.

Finally we find that though there is a difference in the definition of a session, we still find common properties. For example, the number of sessions decreases with the increases of the timeout value.

The daily traffic pattern, weekly traffic pattern, and self-similarity of WAP traffic in this work are found in many other WWW traffics. But we still cannot claim that these properties of WAP in this work are general to all WAP traffics, because we have only analyzed traces of one cellular network.

Chapter 6: Performance Study of the WAP System by LQM

In this chapter, we will study the performance of the WAP system using LQM performance models. The system is simplified as a simple three-layer model and a four-layer model.

6.1 Brief Introduction to LQM

C.U. Smith started the concept of Software Performance Engineering. Since then people have recognized the importance of software performance. Planning the capacity and extension of systems is a familiar concept in industries. Appropriate planning in advance means larger occupation of the market in the future. For example, if a mobile company can predict the increasing trend of the number of customers and the type of services that the customers are going to buy in the future, the company can do capacity and performance studies about the existing systems in advance. By doing so, the company can do necessary adjustments in advance and be well prepared for the enlargement of the customer population and provide high quality services for future customers at reasonable cost. To study the performance of a system, analytical modeling is often used to describe the system. Another approach is to use simulation models. Simulation models can represent a system with more detail that brings the model closer to the real system, but needs more resources and time to obtain the solution. Analytical models often abstract the system with some important characteristics in a mathematical way, neglecting some detail of the system, but they are cheap and can be solved quickly. Queuing Network (QN) is an analytical model that is often used to model computer network systems. QN is

cumbersome in representing software and layered systems. QN has difficulties in describing the contention for a common resource.

Layered Queuing Network (LQN) was developed in the Department of Systems and Computer Engineering at Carleton University. Its aim is to study the performance of distributed systems that have hardware and software. LQN extends QN. LQN is suitable for modeling distributed layered systems, and can include software performance issues easily. LQN has both analytical and simulation tools for solving Layered Queuing Models (LQM). We only use the LQN analytical tool in this work. A task is a basic unit in LQM. A task represents the software in execution. An entry represents a service provided by a task (software). If a task can provide multiple services, then the task has multiple entries. Entries in different tasks communicate with each other through requesting and providing services. For example, a WAP system can be simplified into few client tasks and a server task. In this scenario, the client tasks make requests to the service provided by the server task, and the server task processes the clients' requests and responds to the requests respectively. Requests from clients to the server are either synchronous or asynchronous. A synchronous request means that the request will block the client until it gets the response from the server. An asynchronous request does not block the client.

Next we briefly introduce some concepts, some terminology and notations in LQN. Each task will run on a processor. And each task has at least one entry. The entries in a task have execution demands respectively, and may also interact with entries in other tasks by calling the entries in those tasks. The client tasks will not receive requests from other tasks. They are called reference tasks. For reference tasks, usually there is a think

time that is denoted by Z, which implies the pause time between two consecutive operations. For example, a user starts a Mobile Browser and ends it, after Z seconds, the user starts a Mobile Browser again. The non-reference tasks include tasks that do not make requests (calls) to other tasks and only receive requests from other tasks, and tasks that make requests to other tasks and receive requests at the same time. For execution demands of entries, there are usually two phases. Phase 1 means the serviced call (request) is synchronous, and the entry must provide a response to the call. Phase 2 means the serviced call is asynchronous, and the entry does not provide a response to the call. The LQN analytical tool describes the system by the average behavior of the entries and solves the performance model by approximate MVA calculations. A complete LQN input file and output file of a three layer model are put in Appendix B. More detail about LQN can be found in [Woodside 1995].

6.2 LQM Performance Models

In this section we will give two abstract Layered-Queuing-Model (LQM) models to imitate the system by which a client uses Mobile Browser to access the Internet. Network delay is usually small and we do not consider it in the models. Detailed information about architecture for mobile applications can be found in [Kunz 1999] and [Wang 2000].

Model 1: A Three Layer Model

The first model is shown in Figure 6.2.1. A parallelogram represents an entry of a task. Cascade parallelograms indicate an entry of multiple tasks. The task name is written near the parallelogram. [Z] in the client task entry is the think time of the client. [0, tc] in the

client task entry represents the execution demands of the client task entry. There are two fields inside the brackets, the number in the first field is the execution demand in phase 1, and the second field contains the execution demand in phase 2. Because the client tasks receive no requests from other tasks the execution demand in phase 1 of the client task entry is 0. The pair of brackets inside the non-referential task entries has the same meaning as the one in the client task entry. The notation under the pair of brackets is the entry name. The ellipse represents the CPU processor. The arrow segment connects the calling entry and the called entry. The straight segment connects the task and the CPU on which the task runs. The pair of circular brackets beside the arrow line contains the number of calls from the calling entry to the called entry. ‘sh’ represents synchronous calls and ‘ay’ represents asynchronous calls.

Now we describe the scenarios of the model in Figure 6.2.1. Client tasks make requests to the Gateway server and wait for the responses from the Gateway server. The Gateway server answers some of the requests directly and passes some to other Web servers on the Internet. Generally, passing a request to another Web server takes less time than answering one directly. The Gateway server task plays the role of a client and the role of a server at the same time. It has four entries. The first entry se1 processes the synchronous requests from client tasks and responds to the clients directly. The second entry is responsible for asynchronous requests from the client tasks. The third entry passes synchronous requests from the clients to other Web servers on the Internet and passes back the answers from the Web servers to the clients. The Web server task is used to represent arbitrary Web servers on the Internet since it is impossible for us to get information for all the available Web servers on the Internet and represent each of them.

The fourth entry is used to represent the idle time within a session with the help of an imaginary Idle Server task and CPU4. When a Gateway server is occupied by a session, the only time that the session demands CPU is when requests from the clients within the session are processed. During the idle time of a session, the session does not require the Gateway CPU. The idle time we refer to here is the same as the one that we defined in Section 4.3 and it is a lower bound. But the time interval between two consecutive pairs of request-answer might be longer than the idle time we used here. The capacity of the system obtained with the idle time is a safe bound, which means that the system will not be saturated for sure below that bound. We consider that CPU3 and CPU4 have infinite capacity for simplicity. Each client has a CPU.

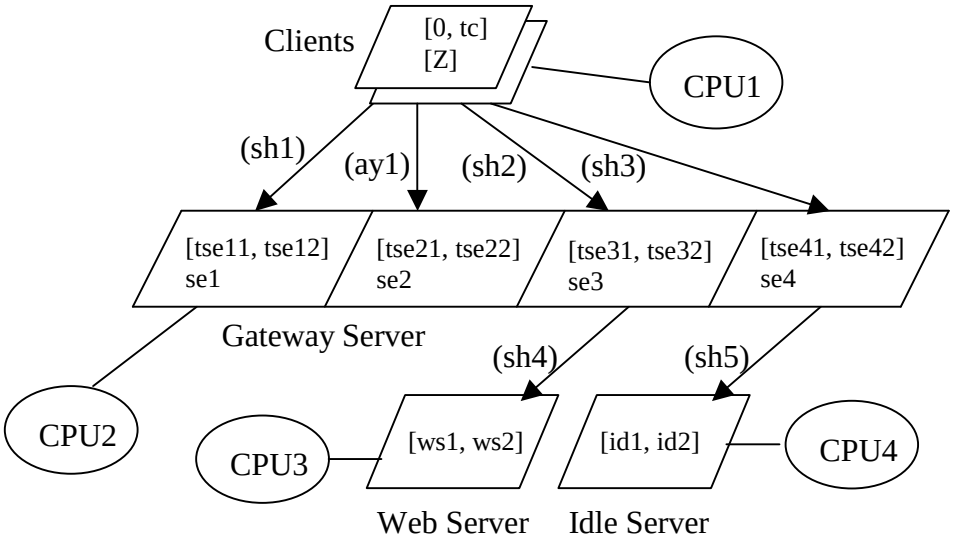

Figure 6.2.1

Model 2: A Four Layer Model

Often there exists a middleware between a client and the gateway server. The middleware is a proxy. The handsets have limited CPU and memory resources. One way to improve

the performance is to pass some load from the handset to a more powerful proxy. To consider this load transition situation we add one proxy task layer below the client tasks (see Figure 6.2.2). We do not have information about the amount of load that is transferred from a handset to a proxy, but we can investigate the relationship between the extent of system improvement and the amount of load that is transferred to the proxy by changing the amount of transferred load.

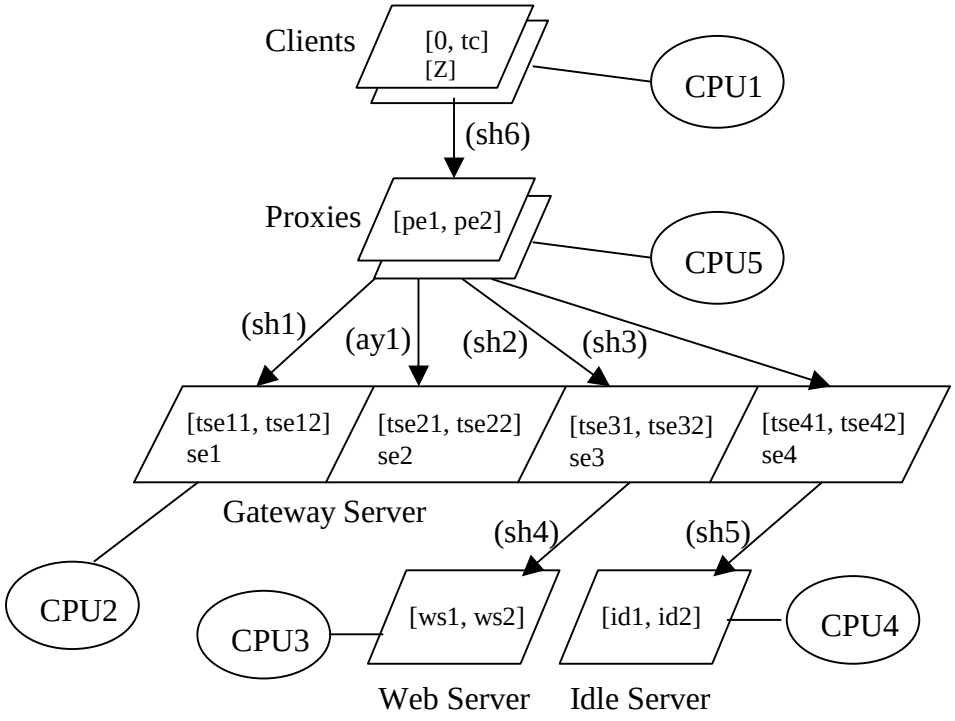

Figure 6.2.2

6.3 Parameters of the Performance Models

In this section we will show how to obtain some parameters for the performance models in Section 6.2 based on the results obtained in Chapter 4. For convenience we repeat some results from Section 4.3 here. Based on the trace files from July 1, 1999 to Dec. 31,

1999, we obtain service time = 0.211 seconds, idle time = 0.856 seconds, 11.19 synchronous requests and 0.766 asynchronous requests per session on average.

Execution Demand:

We have no information about the execution demands on CPU2. But we try to approximate the execution demand by the service time. We assume execution demands for a synchronous request and for an asynchronous request are the same. During the service time CPU2 might be occupied by answering one request directly, maybe more than one request, maybe by a combination of passing some requests to other Web servers and answering some requests directly. The execution demand tse11 should have a lower bound that is equal to the service time since the worst case is that during the service time, the Gateway Server processes only one synchronous request of a session. The upper bound of tse11 could be the value of service time divided by the maximum number of concurrent sessions (MCS) from July 1999 to Dec. 1999. The upper bound tells us that during the service time, the Gateway Server always processes MCS synchronous requests, which will overestimate the power of CPU2. To investigate the lower bound tse11 case, we can give a safe system capacity estimation below which the system will not be saturated for sure. We set tse11 = service time = 0.211 seconds, tse22 = ws1 = tse11, tse12 = ws22 = tse21 = tse32 = tse41 = tse42 = id2 = pe2 = 0, id1 = idle time = 0.856 seconds.

Let R be the relative speed ratio of handset CPU and proxy CPU. If we move x of tc work from the handset to the proxy, then the execution demand in the client is reduced to tc - x, and pe1 = x*R. We assume tc = 6 seconds, tse31 = 0.02 and R = 0.04.

Number of Requests:

The average number of asynchronous requests is 0.766. The average number of synchronous requests is 11.19. For convenience, we approximate 0.766 to 0.77 and 11.19 to 11.2. We will investigate the effect of splitting 11.2 between sh1 and sh2. We set ay1 = 0.77, sh3 = 11.2, sh4 = sh5 = sh6 = 1.

Think Time:

We have no information about the users' behavior. So we do not know the think time. However, we can approximate the range of the think time. The lower bound of the think time is zero, in which case the clients keep sending requests to the Gateway Server at maximum rate. This case will give a safe system capacity estimation. The upper bound of the think time could be one hour that happens during the light traffic period from 2:00AM to 5:00AM. The one-hour think time is meaningless since the light traffic period cannot test the capacity of the system. To estimate the think time practically, we proceed as follows. We consider three numbers during 10:00AM to 5:00 PM (busy traffic period) from July 1, 1999 to Dec. 31, 1999. One number is the average session length denoted as as-len. One number is the average number of concurrent sessions denoted as acs. The last one is the average number of sessions denoted as anum. There are two ways to count the sessions within 10:00AM to 5:00PM. In the first approach, we count a session if either the start or the end of the session is between 10:00AM and 5:00PM. Alternatively, we count a session if both the start and the end of the session are between 10:00AM and 5:00PM. By the first method we have as-len = 1.42 minutes = 85.2 seconds, anum = 687.2. By the second method we have as-len = 1.4 minutes = 84 seconds, anum = 674.1. acs = 2.4 in both methods. The idea to estimate the think time is shown in Figure 6.3.1.

One line segment represents a session, L is the length from 10:00AM to 5:00PM. We have Z = (L – as-len * (anum/acs)) / (anum/acs – 1).

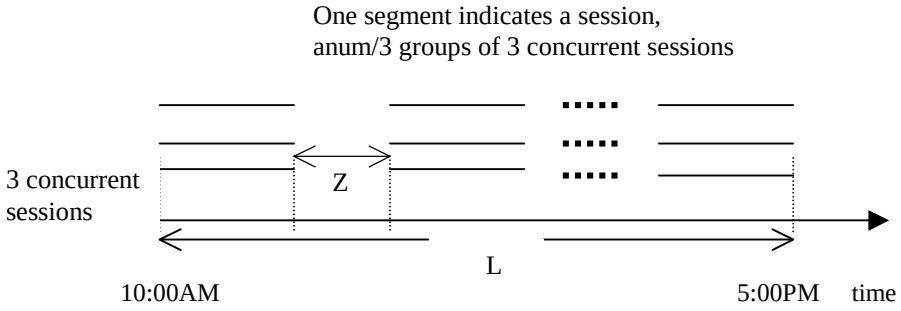

Figure 6.3.1

Method 1 gives Z = 2.8 seconds and method 2 produces Z = 5.7 seconds. We think it is reasonable to use the average (Z = 4.25) to approximate the think time. To investigate the extreme case, we set Z = 0. We will test the effect of increasing the value of Z.

6.4 Results

In this section we will show the capacity of the system under various conditions and the effect of transferring execution load from handsets to proxies. The capacity of the system indicates the maximum number of users that the system can serve at the same time without being saturated. Proxies, Gateway Server, Web Server and Idle Server are multithreaded, and can provide as many threads as needed. We define 0.7 (Bell Mobility uses this number internally) as the threshold utilization of CPU2, beyond which we consider that the system is saturated. ‘MC’ denotes the maximum number of clients that the system can sustain with a utilization of CPU2 below 0.7.

6.4.1 Model 1

The summation of sh1 and sh2 is 11.2, i.e., sh1 + sh2 = 11.2, when sh2 = 1, sh1 = 11.2 – 1 = 10.2, if it is not noted specifically. sh1 = 0 represents the case when all the requests are not directly processed by the Gateway server. sh1 = 11.2 is the case when all the requests are directly processed by the Gateway server. We show the effect of splitting 11.2 between sh1 and sh2 in Table 6.4.1.1. There is only one CPU to support the Gateway server. The capacity of the system decreases with the increase of sh1. That is, the more requests the gateway Server processes directly, the smaller the system capacity becomes. From July 1, 1999 to Dec. 31, 1999, the average maximum numbers of concurrent sessions of each month are below 8, but during the peak hours of some days, the maximum number of concurrent sessions is bigger than 10 (see Appendix A). But the period for the maximum number of concurrent sessions is short, less than 1 minute. This reveals that the system might have short period of the capacity problem from July 1, 1999 to Dec. 31, 1999. In early April 2000, the average maximum number of concurrent sessions has approached the range of values in Table 6.4.1.1, and Bell Mobility reported it had to address capacity problem. This indicates that our models can indeed be useful in anticipating capacity problems.

Table 6.4.1.1 Capacity with different sh1 and sh2

Sh1	0	1.2	2.2	3.2	4.2	5.2	6.2	7.2	8.2	9.2	10.2	11.2
MC	50	36	29	24	20	17	15	13	11	10	9	8

When sh1 = 11.2, the system can serve 8 clients at the same time. We first show the improvement by increasing the speed of CPU2 in Table 6.4.1.2. The speed row in the table contains the relative speedup of the original CPU2.

Table 6.4.1.2 Capacity with different speeds of CPU2

Speed	1	2	3	4	5	10	20
MC	8	13	17	22	26	48	91

We can also improve the capacity by increasing the number of processors that the Gateway Server runs on. Table 6.4.1.3 shows the results. The NUM row indicates the number of processors.

Table 6.4.1.3 Capacity with different numbers of CPU2 (running at the original speed)

NUM	1	2	3	4	5	10	20
MC	8	11	16	21	26	50	100

We can see that both ways can improve the system capacity. But the choice will depend on financial and technical factors. For example, updating the only CPU will cause the shutdown of the system, but adding one more CPU may not. One twice as fast CPU may be much more expensive than two original CPUs.

We also want to know the effect of think time. Let sh1 = 11.2 and other parameters be the same, we change the value of Z. The results are shown in Table 6.4.1.4.

Table 6.4.1.4 Capacity with different think time

Z	0	4	8	12	16	28	40
MC	8	9	11	12	13	17	20

The capacity increases with the increase of think time. We test the effect of the idle time in a session on the capacity too. When we increase the idle time the capacity increases. All these reveal one phenomenon that the system can serve more slow users than fast users as expected. The think time we estimated in Section 6.3 is 4.25. By linear interpolation with the values in Table 6.4.1.4 we find that the capacity at Z = 4.25 is 9.03.

We can say safely that the system can serve 9 clients at the same time without being saturated.

We have little information about what and how many kinds of services the Gateway server provides and the characteristics of these services such as how long one kind of service is on average. The parameters we obtain per session are the average characteristics of all the services together. There are several ways to simulate the different types of services that the system can provide. In the first method, we can use different combinations of sh1 and sh2 to represent different types of services with fixed total idle time within the session. For simplicity, we set sh2 = 0, and only change sh1 to imitate different types of services and see what the capacity is for different services. The results are in Table 6.4.1.5.

Table 6.4.1.5 Capacity versus the increase of sh1 with sh2 = 0 and fixed idle time

Sh1	1	2	3	4	11.2	20	40
MC	16	13	11	10	8	8	7

The capacity decreases with the increase of sh1. This implies that the system can serve more users who use short applications than those who like to use long applications. Long applications result in larger values of sh1, so we can also say that long applications generate heavier traffic than short applications.

In the second method, we increase the request frequency without changing execution demands and the service time of a client. The bound of service time of a client is approximated as ST = 11.2 * (0.211 + 0.856) + 0.77 * 0.211 = 12.11 seconds. We set sh2 = 0 for simplicity. Next we give a group of pairs of sh1 and idle time (sh1=4, idle=0.99), (sh1=8, idle=0.92), (sh1=11.2, idle=0.856), (sh1=16, idle=0.76), (sh1=20,

idle=0.69), (sh1=30, idle=0.5), (sh1=40, idle=0.31), so that ST stays constant. We give the capacity corresponding to the above parameters in Table 6.4.1.6.

Table 6.4.1.6 Capacity with different pairs of sh1 and idle

Sh1	4	8	11.2	16	20	30	40
Idle	0.99	0.92	0.856	0.76	0.69	0.5	0.31
MC	20	22	8	6	5	3	2

We can see that the system capacity decreases sharply with the increase of the request frequency during a fixed period. Comparing the results in Table 6.4.1.6 and Table 6.4.1.5, we find that the capacities at sh1 = 20 and sh1 = 40 in the two tables are very different. This is because the second method generates more intensive traffic than the first method. High frequency requests will result in heavy traffic with high intensity. So the system can serve fewer clients who generate heavier traffic than those who generate lighter traffic, and serve fewer clients who generate the same traffic volume with higher traffic intensity (equivalent to fast users), as expected.

In the third method, we can imitate different services with different execution demands. If the CPU speed is fixed, more execution demand means that heavier traffic data is processed. If the traffic volume is fixed, higher execution demands translate into a relative slower CPU. So we can explain the results in Table 6.4.1.2 from a new angle, i.e., the heavier the traffic is, the lower the capacity of the system is, as expected.

6.4.2 Model 2

Model 2 is a modified version of model 1 by adding a proxy layer. With model 2, we want to investigate the effect of load migration from handsets to the proxies. We use R =

0.04, which means that the proxy CPU is 25 times faster than the handset CPU. The load migration from handsets to the proxies reduces the process time at the clients' side (which is equivalent to making a slow user fast) and therefore increases the traffic. This will reduce the capacity of the system. We set sh1 to 11.2 and sh1 to 0 respectively and see the variation of the capacity with the change in load migration. The results are shown in Table 6.4.2.1 and Table 6.4.2.2.

Table 6.4.2.1 Capacity vs. load migration when sh1 = 11.2

X	0	1	2	3	4	5	6
Tc	6	5	4	3	2	1	0
Pe1	0	0.04	0.08	0.12	0.16	0.2	0.24
MC	8	7	7	7	7	6	6

Table 6.4.2.2 Capacity vs. load migration when sh1 = 0

X	0	1	2	3	4	5	6
Tc	6	5	4	3	2	1	0
Pe1	0	0.04	0.08	0.12	0.16	0.2	0.24
MC	50	46	43	42	41	40	39

We can see that the capacity decreases with the increase of the amount of computational load that is moved from handsets to the proxies. This is consistent with the result in 6.4.1 that shows that the system can serve more slow users than fast ones.

We also investigate how much the performance is improved by load migration. We find that when CPU2 is saturated, load migration hardly improves the performance, but when CPU2 is underutilized, load migration does improve the performance of the system. UT represents the utilization of CPU2 and THPT represents the throughput of

clients. With the conditions of sh1 = 0 and 40 clients in the system we show the load migration effect in Table 6.4.2.3.

Table 6.4.2.3 Performance vs. load migration when sh1 = 0 and 40 clients

X	0	1	2	3	4	5	6
Tc	6	5	4	3	2	1	0
Pe1	0	0.04	0.08	0.12	0.16	0.2	0.24
UT	0.536	0.554	0.572	0.591	0.610	0.617	0.619
THPT	1.39	1.43	1.48	1.53	1.58	1.60	1.61

We find that when the client layer is the bottleneck (larger tc) and the other layers are not saturated, the load migration from handsets to proxies will bring more obvious benefits (bigger throughput and high utilization of underutilized CPU) to the performance of the system.

The test results in this section are obtained under the same condition: the proxy server is multithreaded but runs on only one CPU that is 25 times fast than handset CPU, i.e., all clients are supported by a single proxy machine. Adding one more proxy CPU will not benefit the system since the proxy layer is not the bottleneck with the parameters we used. If we increase the speed of CPU2 greatly, the proxy CPU will become a bottleneck with the increase of clients. We do not show the results here because the tests are somewhat arbitrary: the necessary parameters are not based on the collected trace data. But one thing is certain. If the proxy layer becomes the bottleneck of the system in the future and the other layers are underutilized, it is always possible to improve the performance of the system by adding more proxy CPUs or increasing the speed of the proxy CPU, as we have shown for CPU2.

6.5 Summary

In this chapter we have studied the performance of an abstract WAP system by two LQM performance models. With the test results of model 1 under various circumstances, we find: (1) The system can serve more slow users at the same time than fast users. (2) The system can serve more users who generate light traffic than those who generate heavy traffic. (3) If two kinds of users generate the same amount of traffic volume, the system can serve at the same time more the kind of users who generate traffic with lower intensity than it can do those who generate traffic with higher intensity. (4) By increasing the speed of the Gateway Server CPU or by increasing the number of Gateway Server CPU the system capacity can be improved.

By model 2, we find that load migration from handsets to proxies has the same effect as making slow users faster and the capacity will decrease with the increase of the amount of computation that is moved from handsets to proxies. And load migration from handsets to proxies will benefit more if the client layer is the bottleneck and the other layers are not. The effect of the overhead of load transfer is not depicted in the model, but it can be considered in the model by discounting the amount of load that has been transferred to a certain percentage. Assume the overhead effect is x percentage. If the effect of the overhead of load transfer is included the maximum percentage of load transfer will be $1 - x$. Current applications on the handsets are text based. The processing load on the handset is light and load migration is nit necessary. The aim of the study of load migration is for the future development that may involve heavy load applications on the handsets and proxy layer is necessary.

Chapter 7: Conclusions and Future Work

In this chapter, we will summarize the thesis work in Section 7.1 and discuss possible future work in Section 7.2.

7.1 Conclusions

In this thesis, we studied the WAP traffic of the Bell Mobility cellular network in Quebec and Ontario. We obtained the following main characteristics of the WAP traffic:

- The number of unique IP addresses assigned to clients is linearly increasing with time, after September 6, 1999. It has an obvious week period.
- The traffic increases with the time.
- Average packet (backward packet and forward packet) sizes of each day are constant and time-invariant.
- The ratio of forward traffic volume to backward traffic volume and the ratio of number of backward packets to number of forward packets are constant.
- WAP traffic has obvious day and week periods.
- The traffic is self-similar. H values range from 0.70 to 0.92.

We hope that these characteristics here are general to WAP traffic in other networks. It needs to be verified with more data in other cellular networks. We put forward a layered prediction model that is flexible and easy to use. We also give two definitions of a session, and we think the first definition is more reasonable. With the session model, we have studied the WAP traffic further and some important results are:

- Two versions of session give almost the same results.

- The ratio of backward link activity factor to forward link activity factor is time invariant, though the activity factors themselves change with the time.
- The number of sessions increases with time after September 1999.
- The number of sessions of a day is proportional to the traffic volume of that day. And it has an obvious weekly period.
- The maximum number of concurrent sessions of a day is proportional to the traffic volume of that day.
- 90% of the sessions are less than 4 minutes. This implies that most users like to use short WAP applications.
- The number of longer sessions (s-len > 4 minutes) is a minority, accounting for only 10% of all sessions, but accounting for 34% of the total traffic volume.
- The number of sessions decreases with the increasing of the timeout.

Some results are against intuition. For example, the average backward activity factor is bigger than the average forward activity factor though the forward traffic volume is larger. The longest session is not necessarily the one that has the largest session size. The activity factors do not have a weekly period though repeating peaks tempt people to think so. But spectrum analysis does not reveal a weekly period or any other one.

We also have studied the abstract WAP system with two LQM performance models. The capacity levels of the system obtained from the performance model are in the range of values for the maximum concurrent sessions in late March and April 2000. And Bell Mobility reported that the system experienced capacity problems at that point. So the results from our performance studies have practical relevance. We hope all the work in

this thesis can contribute to the understanding of the Internet usage and WAP systems in general.

7.2 Future Work

There is much scope for future research based on this work. First we hope we can get trace files at other levels and at the client side, so that we can provide a more accurate session model that can describe the user's behavior more closely. We may do some simulation work in the future, so that the traffic analysis can be partly validated. To do simulation, we need a benchmark to generate the network traffic that is as close to real WAP traffic as possible. We do not design a WAP benchmark here, but the results in this work are useful in building such a benchmark. Benchmark design is very important for network simulation, and surely will be one aspect of future work.

With more detailed information of the system available, we can design a more complex and detailed performance model that will be very close to the real WAP system, and that will provide more useful information for the issues of cellular network performance, planning and dimensioning.

It is always important to keep track of new types of the applications that are continuously introduced into the WAP systems with the evolution of the WAP systems themselves. Just like the audio and video traffic brought in different characteristics of the traffic over the networks several years ago, new types of applications may always produce characteristics that are not known before.

References

[Abdulla 1997a] G. Abdulla, et al, *A Realistic Model of Request Arrival Rate to Caching Proxies*, http://vtopus.cs.vt.edu/~chitra/docs/abdulla-nayfeh-fox/paper.pdf, 1997.

[Abdulla 1997b] G. Abdulla, et al, *WWW Proxy Traffic Characterization with Application to Caching*, http://www.cs.vt.edu/~chitra/docs/, 1997.

[Abdulla 1997c] G. Abdulla, et al, *Shared User Behavior on the World Wide Web*, http://www.cs.vt.edu/~chitra/docs/97webnet, 1997.

[Almeida 1998] J. Almeida, P. Cao, *Wisconsin Proxy Benchmark 1.0*, http://www.cs.wisc.edu/~cao/wpb1.0.html, 1998.

[Arlitt 1996] M. F. Arlitt, C. L. Williamson, *Web Server Workload Characterization: The Search for Invariants*, http://www.cs.usask.ca/projects/discus/, 1996.

[Arlitt 1999a] M. F. Arlitt, T. Jin, *Workload Characterization of the 1998 World Cup Web Site*, HP Laboratories Palo Alto, HPL-1999-35, http://www.hpl.hp.com/techreports/1999/HPL-1999-35R1.html, 1999.

[Arlitt 1999b] M. F. Arlitt, et al, *Workload Characterization of a Web Proxy in a Cable Modem Environment*, HP Laboratories Palo Alto, HPL-1999-48, http://www.hpl.hp.com/techreports/1999/HPL-1999-48.html, 1999.

[Barford 1997] Paul Barford, Mark Crovella, *Generating representative Web workloads for network and server performance evaluation*, (also in Proceedings of ACM SIGMETRICS'98), http://www.cs.bu.edu/techreports/97-006-surge.ps.Z, 1997.

[Bell 1999] Bell Mobility, http://www.bellmobility.ca/digitaldata, 1999.

[Catledge 1995] L. D. Catledge, J. Pitkow, *Characterizing Browsing Strategies in WWW*, Computer Networks and ISDN Systems, v26, n6, 1995, pp.1065-1073.

[Choi 1999] Hyoung-Kee Choi, John O. Limb, *A Behavioral Model of Web Traffic*, http://users.ece.gatech.edu/~hkchoi/model.pdf, 1999.

[Comer 1988] Douglas E. Comer, *Internetworking with TCP/IP*, ISBN: 0-13-470154-2, Englewood Cliffs, NJ, Prentice Hall, 1988.

[Crovella 1995] Mark E. Crovella, Azer Bestavros, *Explaining World Wide Web Traffic Self-Similarity*, Department of Computer Science, Boston University, Boston, MA, Technical Report BUCS-TR-95-015, 1995.

[Crovella 1997a] Mark E. Crovella, Azer Bestavros, *Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes*, IEEE/ACM Transactions on Networking, v5, n6, 1997, pp.835-846.

[Crovella 1997b] Mark E. Crovella, et al, *Heavy-Tailed Probability Distributions in the World Wide Web*, in A Practical Guide to Heavy Tails, Adler, et al, Ed., ISBN: 3-7643-3951-9, Birkhauser, Boston, MA, Prentice Hall, 1997.

[Cunha 1995] C. Cunha, et al, *Characteristics of WWW Client-Based Traces*, TR-95-010, Department of Computer Science, Boston University, Boston, MA, http://www.cs.bu.edu/techreports/95-010-www-client-traces.ps.Z, 1995.

[Duska 1997] Bradley M. Duska, et al, *The Measured Access Characteristics of WWW Client Proxy Caches, Dept. of Comp*, Science, U. of British Columbia, TR-97-16, BC, Canada, 1997.

[Garg 1997] Vijay K. Garg, *Applications of CDMA in Wireless/Personal Communications*, ISBN: 0-13-572157-1, Upper Saddle River, NJ, P.H., 1997.

[Gibson 1999] Jerry D. Gibson, *The Mobile Communications Handbook*, ISBN: 0-8493-8597-0, Boca Raton, Fla., CRC Press, 1999.

[Iyengar 1998] Arun K. Iyengar, et al, *Analysis and Characterization of Large Scale Web Server Access Patterns and Performance*, (also in World Wide Web, June 1999), http://www.research.ibm.com/people/i/iyengar/arun2.html, 1998.

[James 1999] E. James, *Summary of WWW Characterizations*, World Wide Web, Jan. 1999, pp.3-13.

[Kunz 1999] Thomas Kunz, et al, *An architecture for adaptive mobile applications*, Proceedings of Wireless 99, the 11th International Conference on Wireless Communications, Calgary, Alberta, Canada, July 1999, pp. 27-38.

[Kunz 2000] Thomas Kunz, et al, *WAP traffic: Description and comparison to WWW traffic*, to appear in Proceedings of the Third ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, (MSWiM 2000), Boston, USA, August 2000.

[Lee 1998] J. S. Lee, L. E. Miller, *CDMA Systems Engineering Handbook*, ISBN: 0-89006-990-5, Boston, Mass., Artech House, 1998.

[Leland 1994] W.E. Leland, et al, *On the Self-similar Nature of Ethernet Traffic*, IEEE/ACM Transactions on networking, v2, n1, 1994, pp.1-14.

[Lam 1997] Derek Lam, et al, *Teletraffic Modeling for Personal Communications Services*, IEEE Communications Magazine, v35, n2, 1997, pp.79-87.

[Mah 1997] B.A. Mah, *An Empirical Model of HTTP Network Traffic*, in Proceedings of INFOCOM'97, Kobe, Japan, April, 1997, pp.7-11.

[Mah 1998] B.A. Mah, et al, *IPB: An Internet Protocol benchmark Using Simulated Traffic*, in Proceedings of the Sixth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'98), Montreal, Canada, July 1998, pp.21-24.

[Mathur 1996] A. Mathur, M. Abrams, *Modeling Trace Data*, TR 96-14, Dept. of Comp. Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1996.

[Mogul 1995] J. C. Mogul, *Network Behavior of a Busy Web Server and Its Clients*, Research Report 95/5, Digital Western Research Laboratory, CA, ftp://gatekeeper.dec.com/pub/DEC/WRL/research-reports/WRL-TR-95.5.ps, 1995.

[Omar 2000] Salim Omar, et al, *Mobile code, adaptive mobile applications, and network architectures*, to appear in Proceedings of the Second International Workshop on Mobile Agents for Telecommunication Applications (MATA'00), Paris, France, September 2000.

[Pahlahvan 1994] K. Pahlahvan, A.H. Levesque, *Wireless Data Communication*, Proceedings of IEEE, v82, n9, 1994, pp.1398-1430.

[Paxson 1995] Vern Paxson, Sally Floyd, *Wide-Area Traffic: The Failure of Poisson Modeling*, IEEE/ACM Transactions on Networking, v3, n3, 1995, pp.226-244.

[Redl 1998] S. H. Redl, et al, *GSM and Personal Communications Handbook*, ISBN: 0-89006-957-3, Norwood, MA, Artech House, 1998.

[Silicon 1996] Silicon Graphics Inc., *World Wide Web Server Benchmarking*, http://mail.mindcraft.com/webstone/ws2-descr.html, 1996.

[SPEC 1996] SPECweb96 *Benchmark*, http://www.specbench.org/org/web96, 1996.

[Tanenbaum 1996] Andrew S. Tanenbaum, *Computer Networks*, ISBN: 0-13-349945-6, Upper Saddle River, NJ, Prentice Hall, 1996.

[Taqqu 1997] M. S. Taqqu, V. Teverovsk, *On Estimating the Intensity of Long-Range Dependence in Finite and Infinite Variance Time Series*, in A Practical Guide to

Heavy Tails, Adler, et al, Ed., ISBN: 3-7643-3951-9, Birkhauser, Boston, MA, Prentice Hall, 1997.

[Thompson 1997] K. Thompson, et al., *Wide-Area Internet Traffic Patterns and Characteristics*, IEEE Network Magazine, v11, n6, 1997, pp.10-23.

[Wang 2000] J. Wang and T. Kunz, *A proxy server infrastructure for adaptive mobile applications*, Proceedings of the Eighteenth IASTED International Conference on Applied Informatics, Innbruck, Austria, February 2000, pp. 561-567.

[WAP 1998] WAP Forum, *WAP Architecture*, Version 30-Apr-1998, http://www.wapforum.com, 1998.

[Woodside 1995] C. M. Woodside, et al, *A Guide to Performance Modelling of Distributed Client-Server Software Systems with Layered Queuing Networks*, SCE95-23, Carleton University, Ottawa, Canada, November 1995.

Appendix A: Additional Results of Session Analysis

In this appendix, first we present session results for August, Sept., Oct., Nov. and Dec. 2000 in Figures A.1 – A.25.

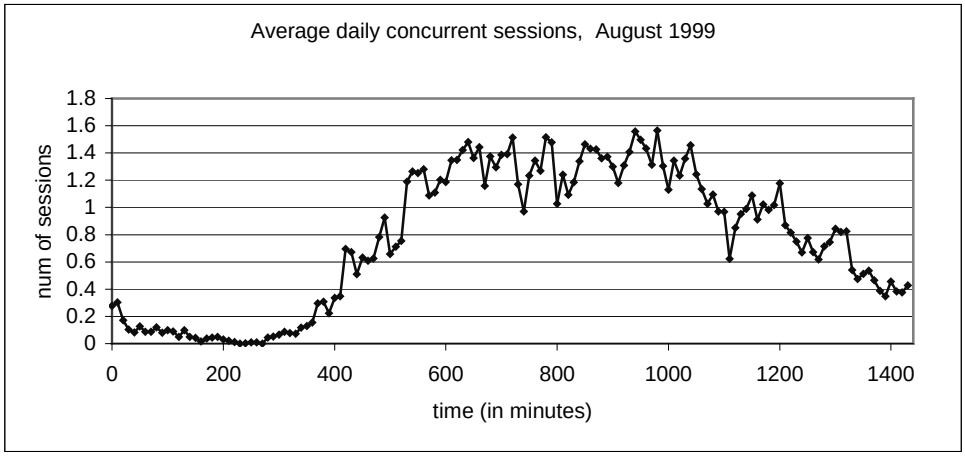

Figure A.1

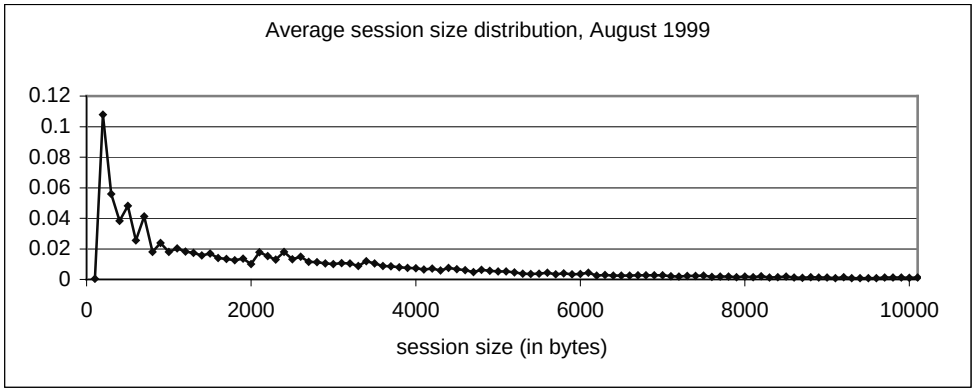

Figure A.2

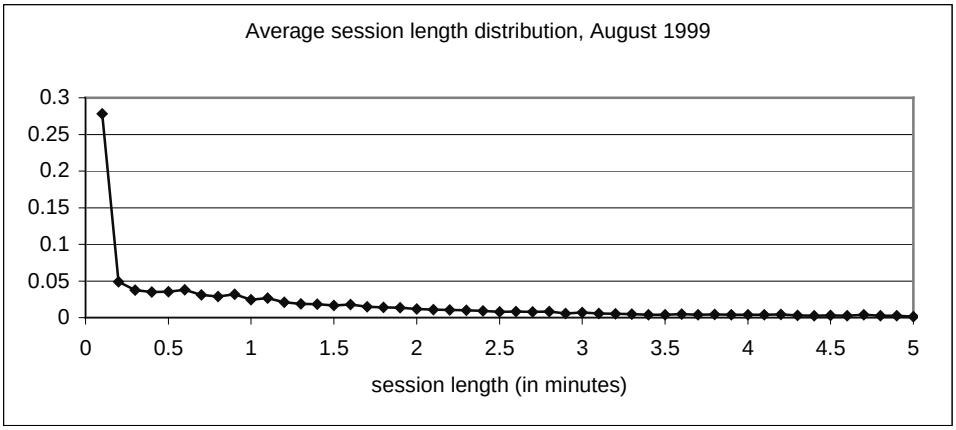

Figure A.3

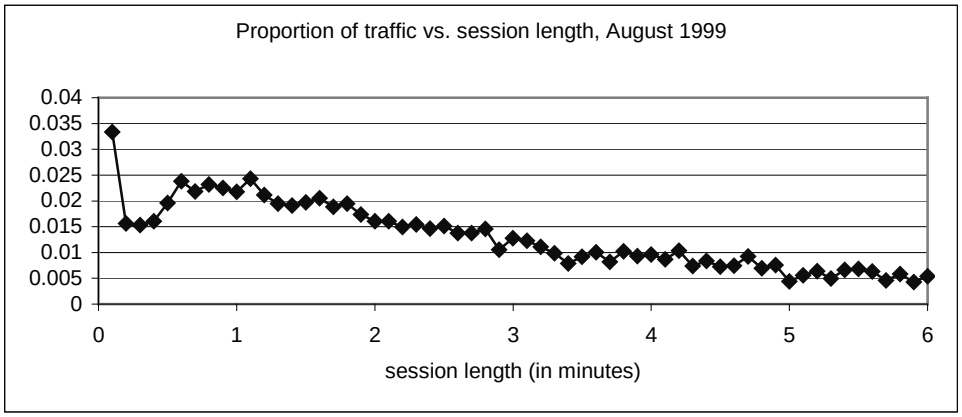

Figure A.4

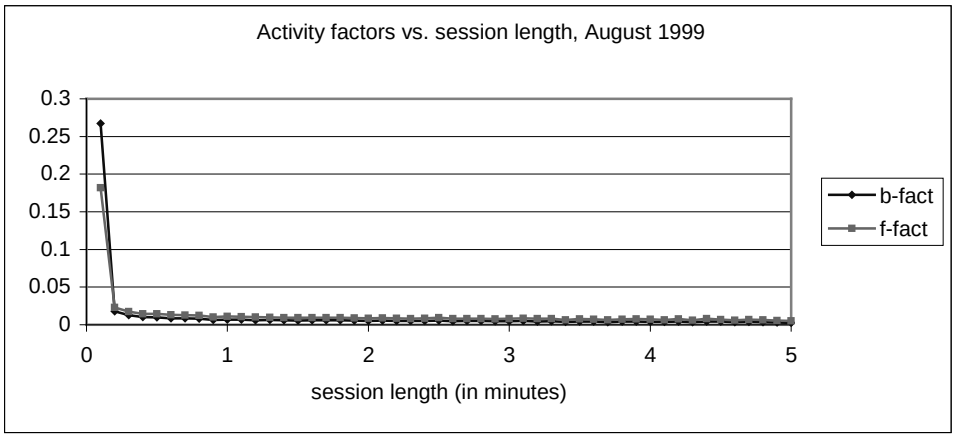

Figure A.5

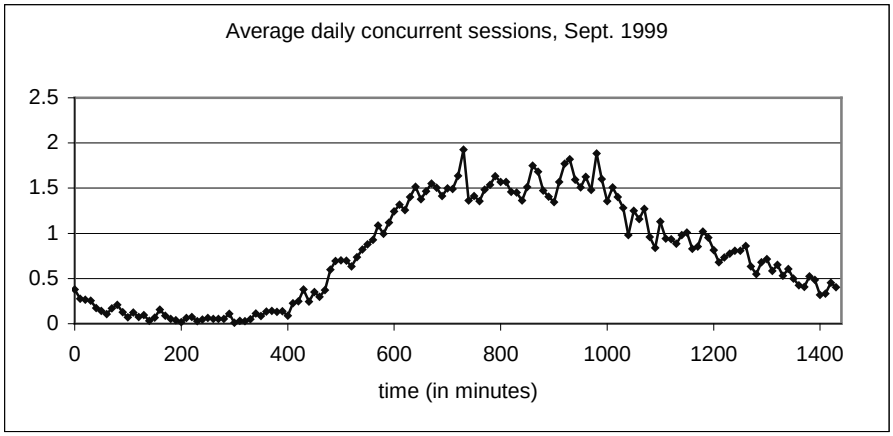

Figure A.6

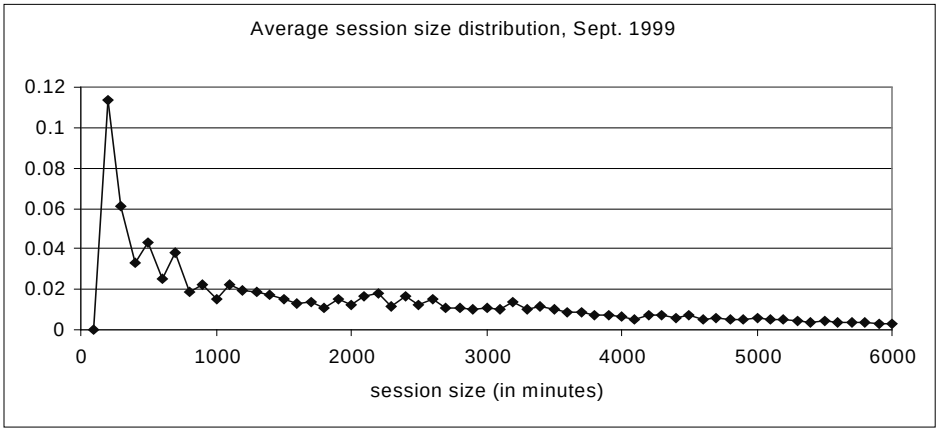

Figure A.7

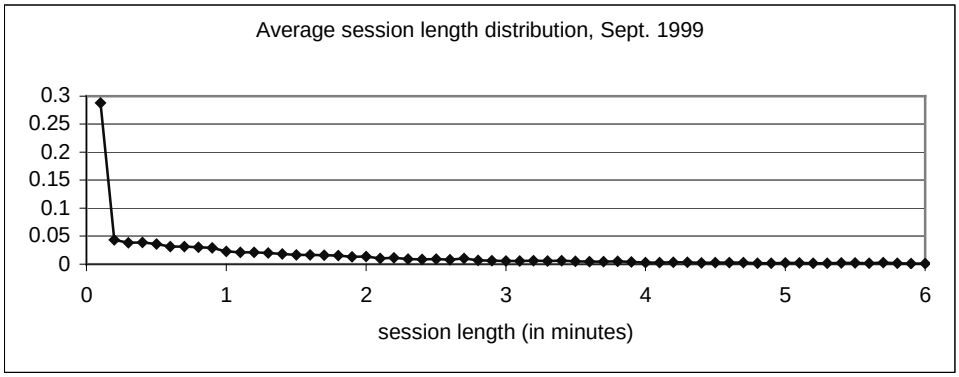

Figure A.8

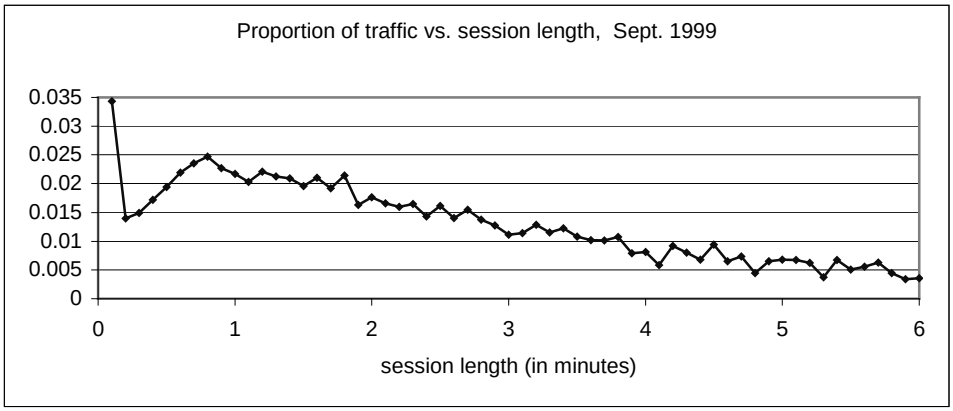

Figure A.9

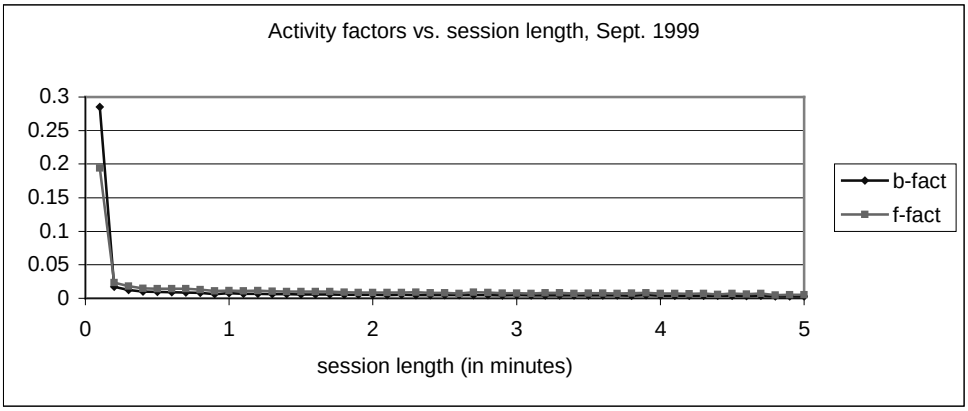

Figure A.10

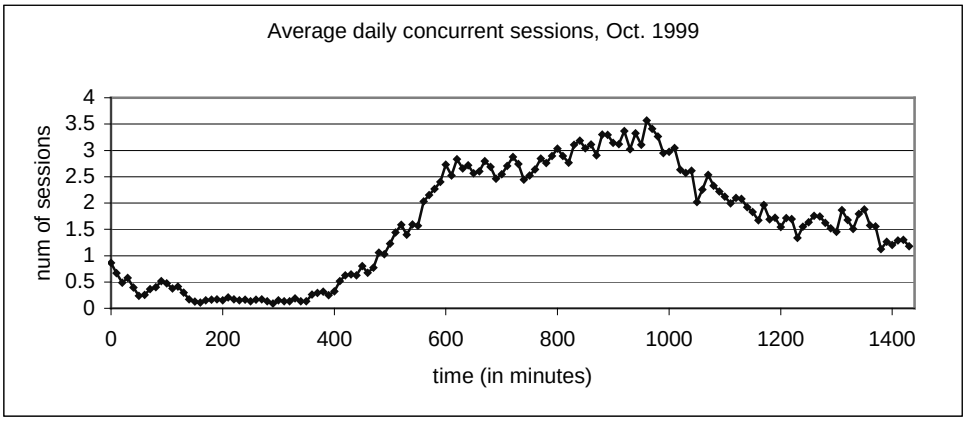

Figure A.11

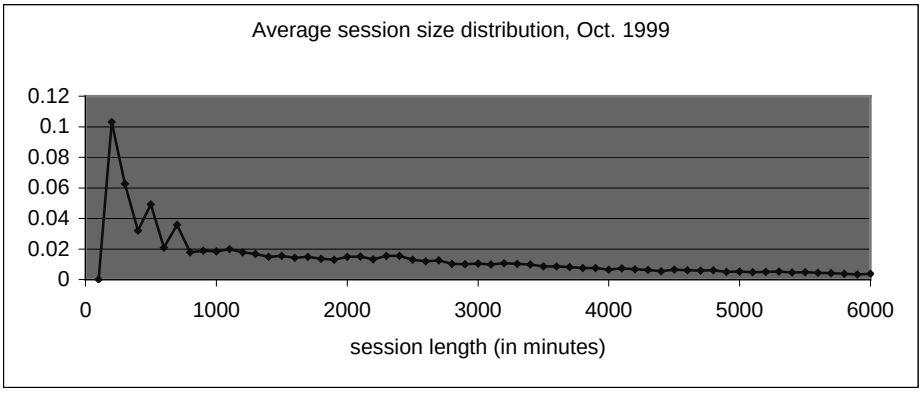

Figure A.12

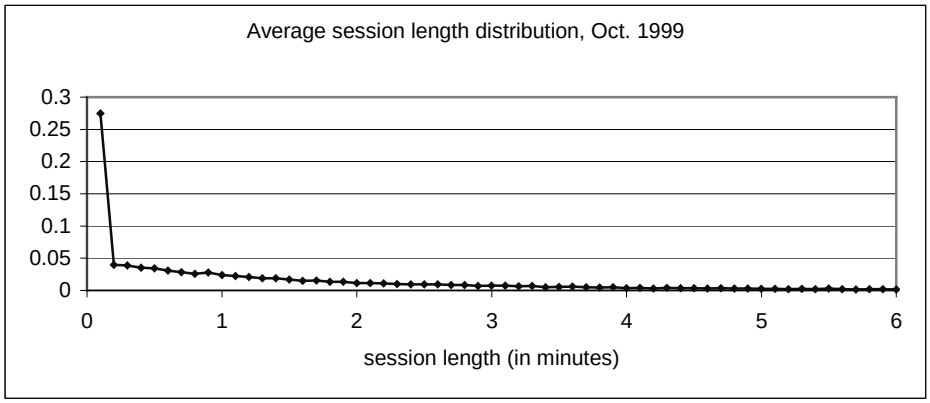

Figure A.13

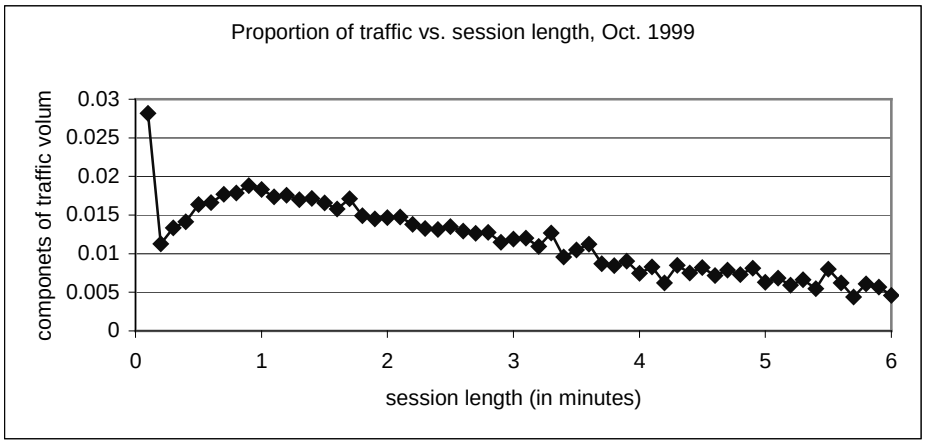

Figure A.14

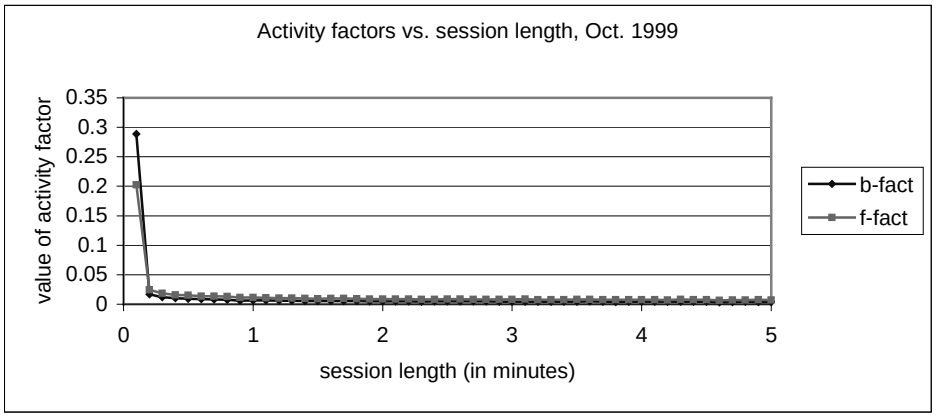

Figure A.15

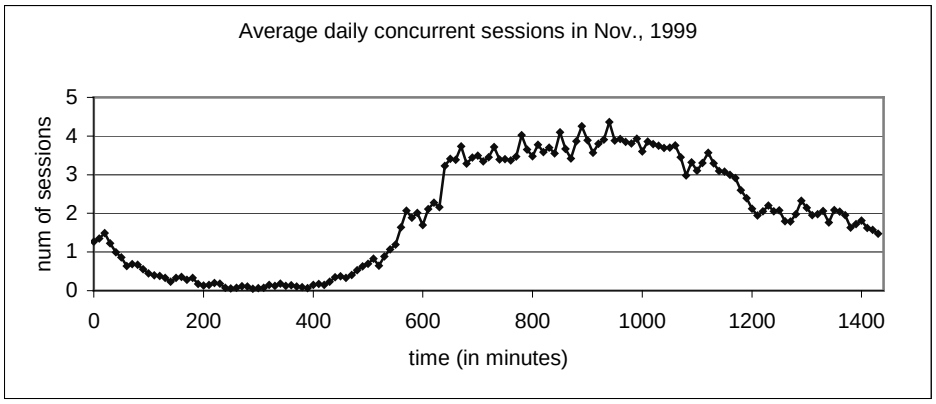

Figure A.16

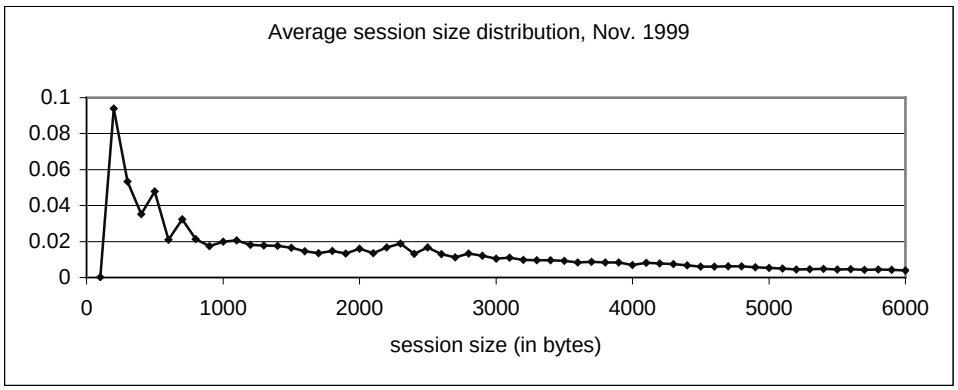

Figure A.17

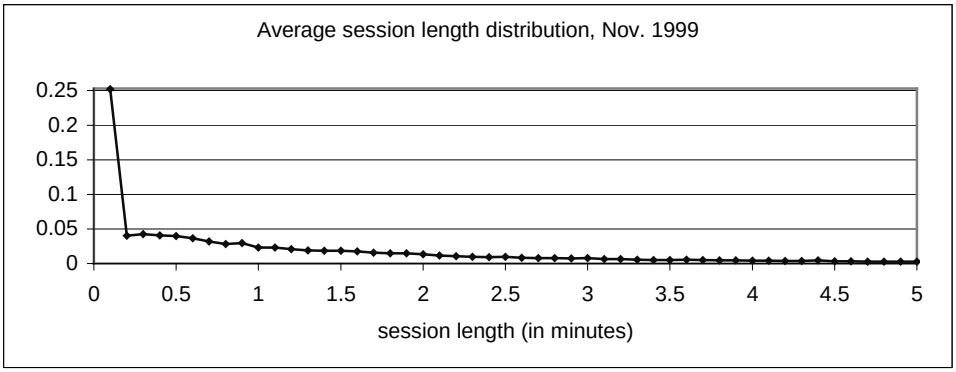

Figure A.18

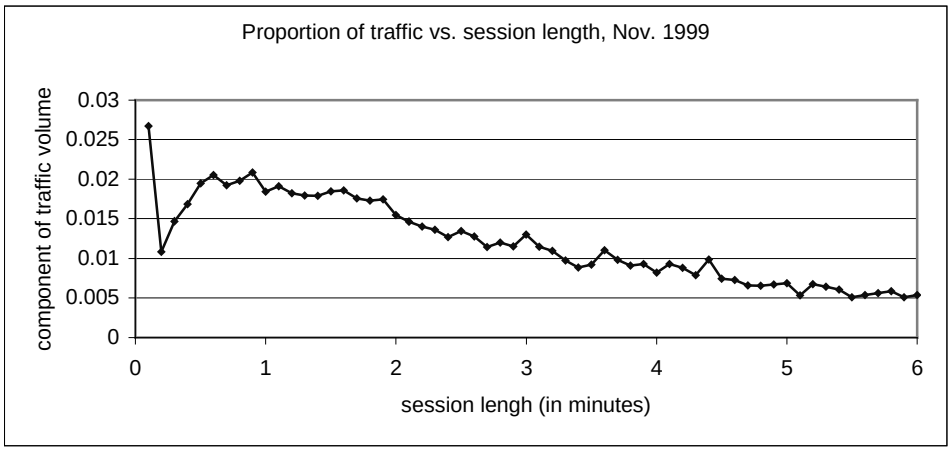

Figure A.19

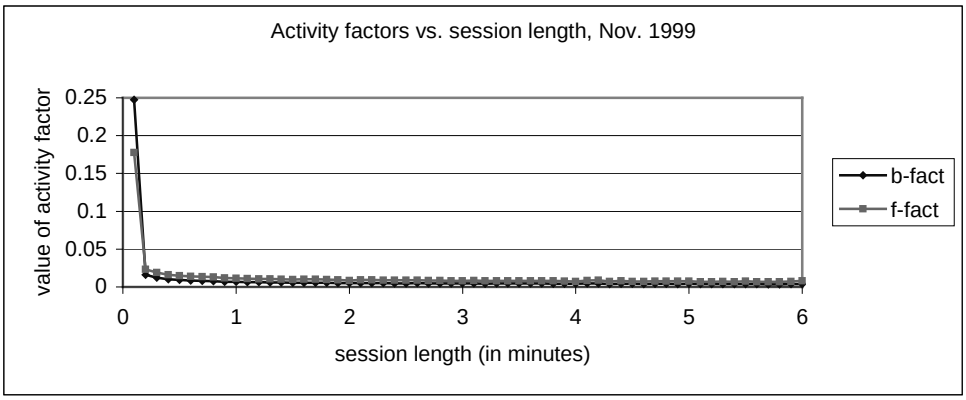

Figure A.20

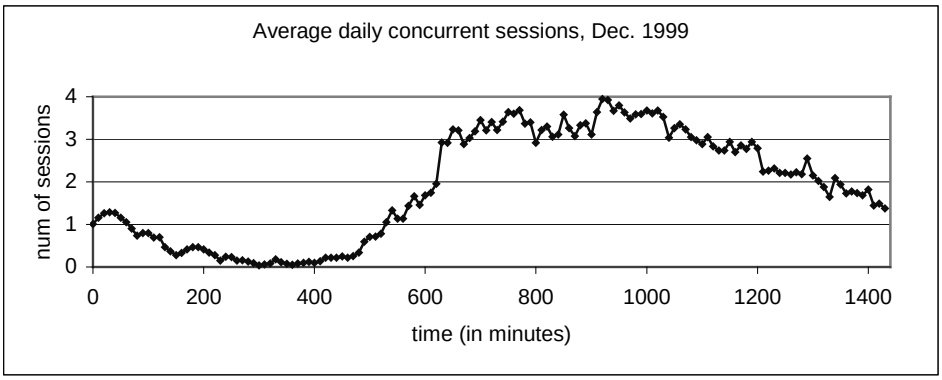

Figure A.21

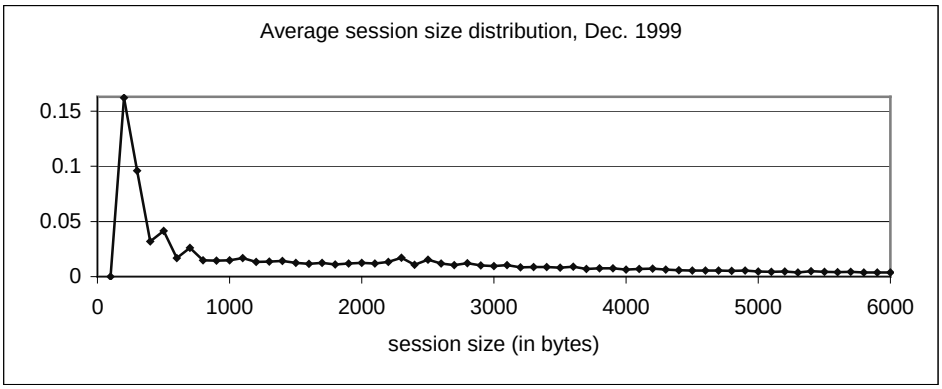

Figure A.22

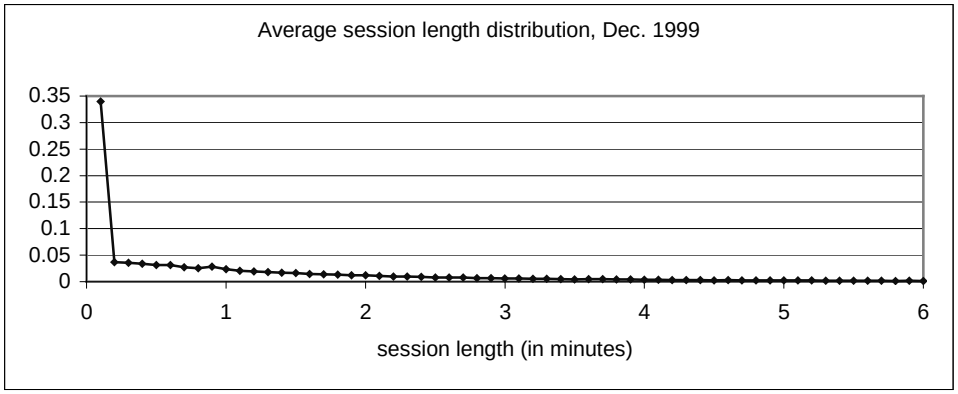

Figure A.23

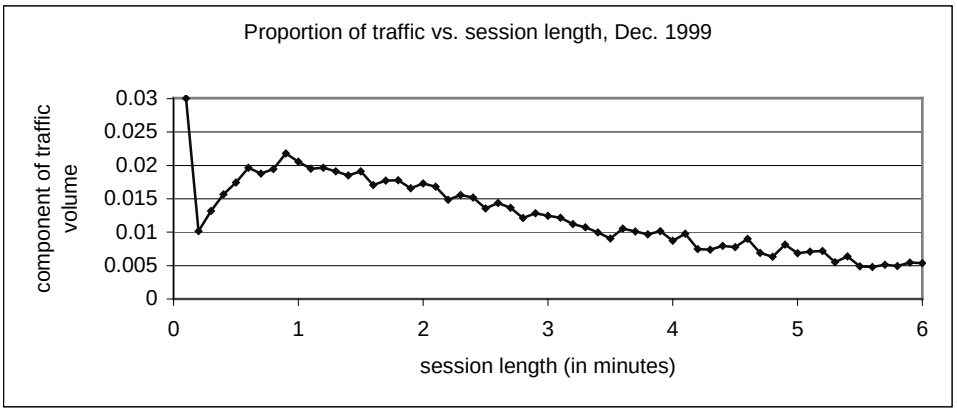

Figure A.24

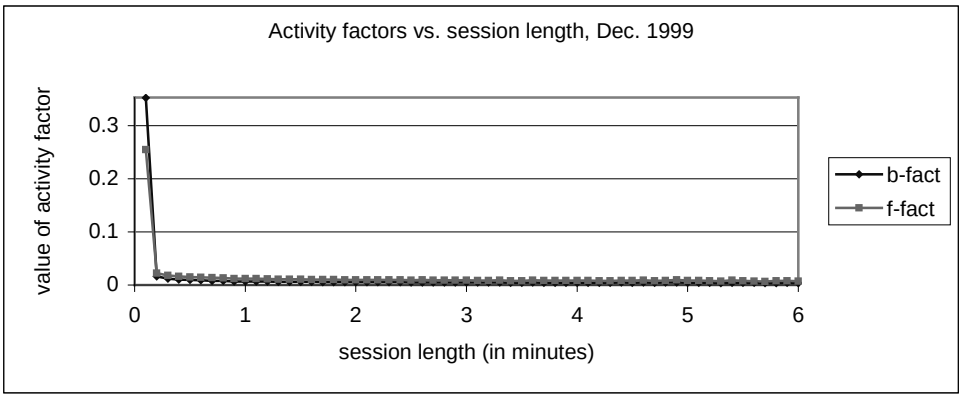

Figure A.25

The number of sessions per day from July 1, 1999 to Dec. 31, 1999 is shown in Figure A.26. The maximum number of con current sessions each day from July 1, 1999 to Dec. 31, 1999 is shown in Figure A.27. Please note that the maximum number of concurrent sessions here is not an average result like the one in Figure A.1. Both figures show a clear week period (which is also confirmed by the spectral analysis). The influence of Christmas is reflected by a decreasing trend at the end of December.

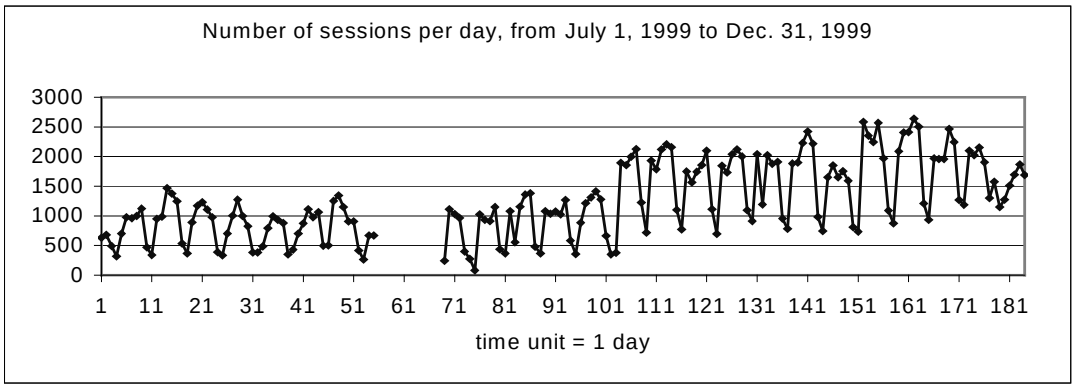

Figure A.26

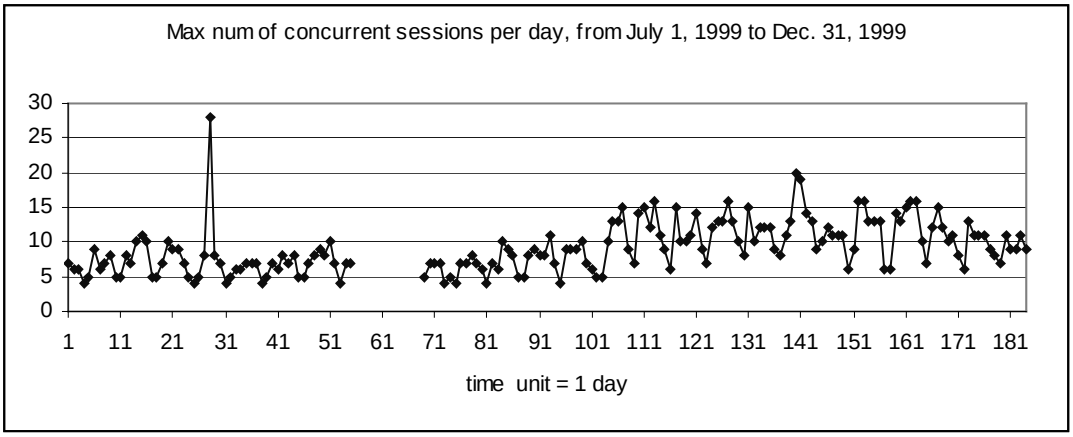

Figure A.27

Now we like to say something more about the concurrent sessions. During a day, the period of 0:00 – 24:00 is divided with a bin of *b* minutes. In each bin, e.g., the *n*th bin, if a session overlaps with the bin, we count this session at time *n*. If there are *x* sessions overlap with the bin, we say that at time *n* the number of concurrent sessions is *x*. The smaller the bin size *b* is, the smaller the number of concurrent sessions will become as

expected. But there is a limit. When *b* is less than the limit, the number of concurrent sessions no longer decreases with the decreasing of *b*. We have found the limit in this work is 0.1 minutes. To draw the figure about the number of concurrent sessions in a day, 24 * 60 * 10 = 14400 points are needed. The data is too big. So we use the average of 100 points to reduce the data size. We can also use the maximum one of the 100 points, but we think the average can represent the 100 points better than the maximum point. All figures about the number of concurrent sessions per day from July to Dec. 2000 are drawn in this way. The problem of the method is that the maximum number of the concurrent sessions per day will be reduced by the average. After average the maximum numbers of concurrent sessions from July to Dec. are 1.8, 1.6, 1.9, 3.6, 4.4 and 4.0. Before average the maximum numbers of concurrent sessions from July to Dec. are 2.2, 2.2, 2.5, 4.3, 5.0 and 4.6. The average number of concurrent sessions over a day from July to December are 0.89, 0.77, 0.79, 1.62, 1.97 and 1.89. The Christmas in December makes the values in December smaller than those in November.

Appendix B: An Example of LQN Input and Output Files

In this appendix, we show that how the LQN input and output files look like for Model 1.

```
G
#This is a LQN input file
#Comments between quotes, as many lines as necessary
"Simple client-server model, only three layers, server has 4 entries"
#Convergence criterion, iteration limit, print interval, under-relax
#Under-relaxation coefficient stabilizes the algorithm if less than 1
#entry se1 represents synchronous requests
#entry se2 represents asynchronous requests
#entry se3 represents passing requests to other web servers
#entry se4 represents the idel time in a session
0.00001
200
1
0.9
# End of General Information
-1

# Processor Information: No of processors
P 4
#p ProcessorName SchedDiscipline [multiplicity, default = 1]
#         Discipline = f fifo|r random|p premptive|
#                              h hol or non-pre-empt|s proc-sharing
#         multiplicity = m value (multiprocessor)|i (infinite)
p c1 f i
p s1 f
p s2 f i
p s3 f i
# End of Processor Information
-1

# Task Information: No of Tasks
T 7
#t TaskName RefFlag EntryList -1 Processor [multiplicity]
#         RefFlag = r (reference or user)|n (other)
#         multiplicity = m value (multiprocessor)|i (infinite)
T 4
t tc1 r ce1 -1 c1 m 8
t ts1 n se1 se2 se3 se4 -1 s1 m 8
t ts2 n ws1 -1 s2 m 10
t ts3 n id1 -1 s3 m 10
# End of Task Information
-1

#Entry Information: No. of Entries
```

```
E 7
# ParameterToken EntryName Phase1value Phase2 Phase3 -1
# Tokens and Value Definitions are:
# s HostServiceRequests for EntryName
# c HostServiceCoeeficientofVariation
# f PhaseTypeFlag
# These lines go Token FromEntry ToEntry Phase1Value Phase2 Phase3 -1
# Tokens and Value definitions are:
# y SynchronousCalls (no. of rendezvous)
# F ProbForwarding (forward to this entry rather than replying)
# z AsynchronousCalls (or no. of sned-no-reply messages)
# o Fanout (for replicated servers)
# i FanIn (for replicated servers)
s ce1 0 6 0 -1
#Z ce1 0 4.25 0 -1
y ce1 se1 0 11.1999 0 -1
y ce1 se3 0 0.0001 0 -1
y ce1 se4 0 11.2 0 -1
z ce1 se2 0 0.77 0 -1
s se1 0.211 0 0 -1
s se2 0 0.211 0 -1
s se3 0.02 0 0 -1
y se3 ws1 1 0 0 -1
s se4 0.000001 0 0 -1
y se4 id1 1 0 0 -1
s id1 0.856 0 0 -1
s ws1 0.211 0 0 -1
#End of Entry Information
-1
```


Generated by lqns, version 28.2 (UNIX)
Input: mthree.lqn
Output: mthree.out
Comment: Simple client-server model, only three layers, server has 4 entries

Convergence test value: 3.82591e-06
Number of iterations: 8

MVA solver information:

Layer	n	step()	mean	stddev	wait()	mean	stddev
1	8	45	5.625	0.18298	765	95.625	6.0384
2	15	190	12.667	0.984	10440	696	137.47
3	8	76	9.5	0.18898	5792	724	28.725
Total	31	311	10.032	0.71012	16997	548.29	81.849

sunrise.sce.carleton.ca SunOS 5.7

User: 0:00:00.12
System: 0:00:00.01
Elapsed: 0:00:00.16

Processor identifiers and scheduling algorithms:

Processor Name	Type	Copies	Scheduling	
c1	Inf	1	DELAY	
s1	Uni	1	FCFS	(V phases)
s2	Inf	1	DELAY	
s3	Inf	1	DELAY	

Task information:

Task Name	Type	Copies	Processor Name	Pri	Entry List
tc1	ref(8)	1	c1	0	ce1 (2 phases)
ts1	mult(8)	1	s1	0	se1, se2, se3, se4 (2 phases)
ts2	mult(10)	1	s2	0	ws1
ts3	mult(10)	1	s3	0	id1

Entry execution demands:

Task Name	Entry Name	Phase 1	Phase 2
tc1	ce1	0	6
ts1	se1	0.211	0
	se2	0	0.211
	se3	0.02	0
	se4	1e-06	0
ts2	ws1	0.211	0
ts3	id1	0.856	0

Mean number of rendezvous from entry to entry:

Task Name	Source Entry	Target Entry	Phase 1	Phase 2
tc1	ce1	se1	0	11.1999
	ce1	se3	0	0.0001
	ce1	se4	0	11.2
ts1	se3	ws1	1	0
	se4	id1	1	0

Mean number of non-blocking sends from entry to entry:

Task Name	Source Entry	Target Entry	Phase 1	Phase 2
	ce1	se2	0.77	

Phase type flags:
All phases are stochastic.

Squared coefficient of variation of execution segments:
All executable segments are exponential.

Open arrival rates per entry:
All open arrival rates are 0.

Type 1 throughput bounds:

Task Name	Entry Name	Throughput
tc1	ce1	0.055709
ts1	se1	4.73934
	se2	4.73934
	se3	4.329
	se4	1.16822
ts2	ws1	4.73934
ts3	id1	1.16822

Mean delay for a rendezvous:

Task Name	Source Entry	Target Entry	Phase 1	Phase 2
tc1	ce1	se1	0	0.00742656
	ce1	se3	0	0.0121277
	ce1	se4	0	0.0208157
ts1	se3	ws1	0	
	se4	id1	0	

Service times:

Task Name	Entry Name	Phase 1	Phase 2
tc1	ce1	0	28.4813
ts1	se1	0.520386	0
	se2	0	0.520386
	se3	0.849771	0
	se4	1.45863	0
ts2	ws1	0.211	0
ts3	id1	0.856	0

Service time variance (per phase)
and squared coefficient of variation (over all phases):

Task Name	Entry Name	Phase 1	Phase 2	coeff of var **2
tc1	ce1	0	882.928	1.08844

ts1	se1	0.044521	0	0.164405
	se2	0	0.044521	0.164405
	se3	0.629599	0	0.871887
	se4	3.69954	0	1.73884
ts1	Total	2.03145		
ts2	ws1	0.044521	0	1
ts3	id1	0.732736	0	1

Throughputs and utilizations per phase:

Task Name	Entry Name	Throughput	Phase 1	Phase 2	Total
tc1	ce1	0.280886	0	8	8
ts1	se1	3.1459	1.63708	0	1.63708
	se2	0.216281	0	0.11255	0.11255
	se3	2.80886e-05	2.38689e-05	0	2.38689e-05
	se4	3.14593	4.58873	0	4.58873
	Total	6.50813	6.22583	0.11255	6.33838
ts2	ws1	2.80887e-05	5.92671e-06	0	5.92671e-06
ts3	id1	3.14593	2.69292	0	2.69292

Utilization and waiting per phase for processor: c1

Task Name	Pri	n	Entry Name	Utilization	Ph1 wait	Ph2 wait
tc1	0	8	ce1	1.68532	0	0

Utilization and waiting per phase for processor: s1

Task Name	Pri	n	Entry Name	Utilization	Ph1 wait	Ph2 wait
ts1	0	8	se1	0.663784	0.309385	0
			se2	0.0456353	0	0.309385
			se3	5.61772e-07	0.309385	0
			se4	3.14593e-06	0.309385	0
ts1	Total			0.709423		

Utilization and waiting per phase for processor: s2

Task Name	Pri	n	Entry Name	Utilization	Ph1 wait	Ph2 wait
ts2	0	10	ws1	5.92671e-06	0	0

Utilization and waiting per phase for processor: s3

Task Name	Pri	n	Entry Name	Utilization	Ph1 wait	Ph2 wait
ts3	0	10	id1	2.69292	0	0