Computer Science

Jessica Gustavsson

Mari Göransson

An Evaluation of

Wireless Application Protocol Traffic

Bachelor's Project

2000:08

An Evaluation of

Wireless Application Protocol Traffic

Jessica Gustavsson

Mari Göransson

This report is submitted in partial fulfillment of the requirements for the Bachelor's degree in Computer Science. All material in this report which is not our own work has been identified and no material is included for which a degree has previously been conferred.

Jessica Gustavsson

Mari Göransson

Approved, 05-30-2000

Advisor: Anna Brunström, KaU

Examiner: Stefan Lindskog, KaU

Abstract

This report is aimed towards presenting, and to a certain degree explaining, the current Wireless Application Protocol (WAP) traffic situation. WAP is a fairly new area within telecommunications and it is important for developers to have accurate and precise traffic models in order to foresee feasible traffic patterns. This report first provides basic knowledge about the WAP structure and its functionality, and then focuses on evaluating WAP traffic.

The concepts of traffic models are also introduced, including why they are used, how they are developed and some important aspects to consider when modeling traffic. Our traffic modeling is directed towards investigating the wireless application protocol users and the traffic they produce, specifically parameters showing daily usage and how that usage varies. Theories concerning the traffic situation are presented and explored through graphs showing important features.

By understanding the characteristics of a system's workload, with the help of for instance traffic models, it is easier to make performance improvements. This is why the intended audience of this report is WAP developers as well as graduate students.

Acknowledgements

We would like to extend our most sincere gratitude towards the following individuals who have contributed in different and highly appreciated ways in the making of this report:

Mikael Nilsson

- For his patient explanations, advise and humorous guidelines (when we side stepped from our assignment).

Anna Brunström

- For her commitment to this assignment, for her thorough perusals and advise regarding improvements and different approaches.

Admir Vegar

- For his assistance and advise when technical difficulties arose.

Magnus Ljung and other employees at Ericsson Software Technology AB involved

- For providing logged data from the Wireless Application Protocol Gateway/Proxy in Karlskrona, and for their quick responses to our questions. Without your cooperation, this paper would not exist.

Ingemar Adolfsson

- For gracefully trying to help with our statistical analysis.

Magnus Lindström and Helena Lindskog

- For keeping themselves and us updated on important information concerning our assignment.

Contents

List of Figures

List of tables

1 Introduction

Internet services, such as e-mail, news, and currency updates, are of large interest for an everyday user of the Web. Until recently, these services have been bound to desktop and laptop computers. Today, users can access Internet services through their mobile telephones, pagers, personal digital assistants (PDAs), and other wireless terminals. This development of wireless services has become a reality thanks to the cooperation of the leading companies within telecommunications. In 1997 Ericsson [28], Nokia [22], Motorola [20] and Phone.com [25] created the Wireless Application Protocol (WAP) Forum [30], with the purpose to ensure a single open standard among developed products. The Wireless Application Protocol is about modifications of current Internet standards and techniques to match the demands of wireless networks. Where needed, extensions to the WAP specification are developed and becoming new standards.

Today there are no existing dimensioning tools for estimating the different forms of equipment, for instance server equipment, needed to handle the number of user requests from wireless terminals. Therefore, developers are forced to make approximations based on experience and intuition. An investigation of the wireless traffic situation may be useful and this report is an attempt to clarify whether this situation has some common denominators. In order to create a traffic model a thorough inspection of the WAP traffic functionality is called for. Information on the subject is retrieved from logs in existing systems, logs being files containing data from the generated traffic. The basic assignment in this report is to explore the traffic produced by WAP device users.

As the concept of the Wireless Application Protocol is relatively new, Section 2 is directed towards providing readers with basic knowledge about the WAP technology. It briefly explains why the technology exists today, the purpose of WAP and its architecture. To enlighten the reader, a comparison between the Internet model and the WAP model is performed.

Section 3 introduces the concept of traffic models. The section presents answers to questions such as why traffic models are created and what use developers can have of them. Different ways of collecting the information necessary to create a traffic model are also mentioned. The need to simulate a system’s workload is an important factor when

dimensioning the components of a system and therefore, the two main approaches to generate a synthetic workload are presented.

To give the reader a view of possible statistical distributions that could perhaps be applied to this kind of traffic, Section 4 has the task of summarizing and introducing a few distributions and their typical curves. A deeper mathematical explanation of the distributions is left for Appendix B.

The following section (Section 5) aims towards describing the measurements of WAP traffic, both how and why they were performed. The section will examine the appearance of the used traffic logs and the exact course of action used when performing the assignment of this report. The methodology contains a thorough walkthrough of how we gathered information about the subject of traffic models, from where we obtained logged data, the extraction of important information from the logs and our analysis of the data.

Section 6 is dedicated to our own research, that is, our modeling results. First, a brief overview of the modeling assignment and the background to our model analysis is given, and then we present our ideas and theories on how to find patterns in the WAP traffic. When analyzing our results we give some example graphs, and try to explain their appearance. In order to find some general characteristics in the WAP traffic we observe the interesting parameters from various point of views. As a completion to this section we try to summarize our results and present a general picture of the current WAP traffic situation in our dataset.

In Section 7 we give some improvement suggestions that could be applicable in future examination of WAP traffic. In this report we have chosen one way to evaluate the WAP traffic, however, several other solutions exist and some of them are presented in Section 8. Problems that arose during this project are mentioned in Section 9: the two main problem areas being information gathering and traffic model development. To conclude this report, Section 10 contains final conclusions of this assignment.

2 An Introduction to the Wireless Application Protocol

This section is an attempt to give the reader an insight in a relatively new area of data/telecommunications, namely the Wireless Application Protocol (WAP). In order to follow the development of the requested traffic model, it is essential that the reader understands the characteristics of this environment. Further information about WAP can be found in “W@P White Paper” [33], “WAP White Paper” [3], “Wireless Application Protocol Architecture Specification” [31], “Wireless Application Protocol Wireless Application Environment Overview” [32] and “PC+” [12,18].

2.1 Background

As the Internet market grew larger the leading companies in telecommunications worldwide found a need to access Internet services from the mobile networks, that is, in wireless environments. In 1997 Ericsson [28], Nokia [22], Unwired Planet (now Phone.com) [25] and Motorola [20] joined together and founded the WAP Forum [30]. By allowing the forum to introduce a single open standard a creative development started in wireless technology and since the WAP specifications are closely related to the Internet standards, developers have been able to make the transition to the new wireless technology with ease.

2.2 WAP Forum

Based upon the well documented and proven Internet standards the WAP Forum has released a global wireless protocol specification for the majority of the existing digital wireless networks. One of the main goals of the WAP Forum is to ensure that newly developed standards remain compatible and independent of existing wireless network standards. The WAP specification will not prevent competition between the companies within telecommunications; the open standard will only assure a common development platform. It is applications, such as weather forecasts, currency and mail applications, that will make the products unique. Membership in WAP Forum is open to all organizations and the forum had more than three hundred members as of January 2000.

2.3 Why the Market Needs WAP

There are a few differences between the wired and the wireless networks and between the devices used to connect to them. This is why existing Internet standards cannot be used in wireless networks. Wireless data networks tend to have less bandwidth, higher latency, less connection stability and less predictable availability than wired networks. As shown in Figure 2.1, the wireless devices have limitations such as less powerful CPUs, less RAM and ROM memory, restricted power consumption (limitations of battery life), smaller displays and different input devices (e.g. phone keypad) in comparison to the usual desktop computer.

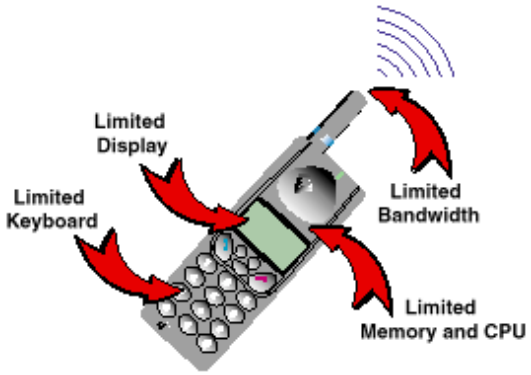


Figure 2.1: The Limitations of a Wireless Device

The typical user of handheld devices, such as mobile telephones, pagers, personal digital assistants (PDAs) and other wireless terminals, has different demands than the user of, for instance, a desktop computer "surfing" the Internet.

Wireless devices must be as easy to use as possible, and the limitations and demands discussed earlier in this section must be set and met, in order to deliver a satisfactory user experience. The WAP specification makes use of the foremost fundamentals of Internet standards and has made some improvements to suit the wireless environment.

2.4 The Internet Model vs. the WAP Model

The WAP model deals with modifications and reuse of existing protocols and techniques. There are clear similarities between the architecture of the Internet model and the WAP model (see Figure 2.2). The protocols in the WAP model will be briefly discussed later in Section 2.5.

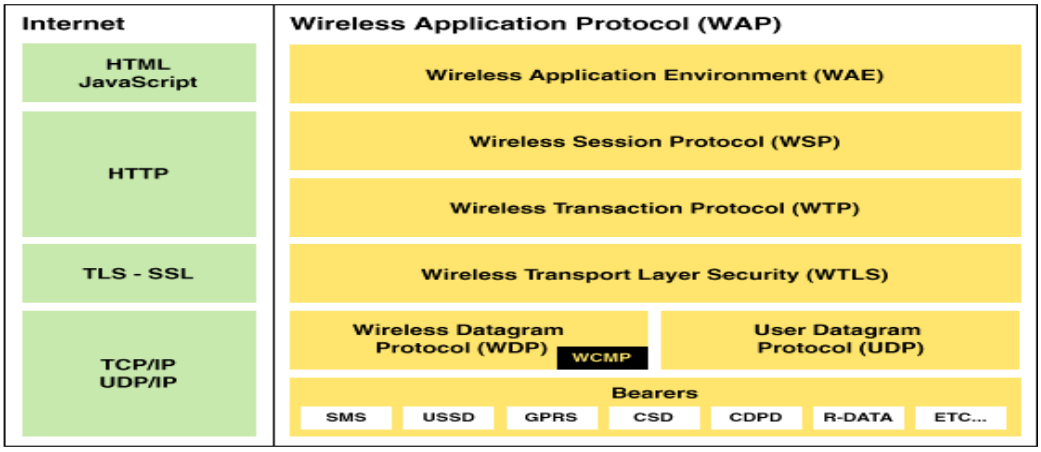


Figure 2.2: The Internet Model versus the WAP Model

In the Internet model a client can connect to various servers in order to reach a broad spectrum of Web services. A HyperText Transfer Protocol (HTTP) request asks for a unique Uniform Resource Locator (URL) from a client, the URL is then sent to a Web server who responds by sending back the requested Web page. This course of events is represented in Figure 2.3.

The content on the Web server is mostly in HyperText Markup Language (HTML) format, but for the dynamic content (where information is updated often) there are also Common Gateway Interface (CGI) scripts and Java Servlets.

A large amount of information is sent and received via the Internet, mostly text based, resulting in a slower connection not acceptable in the wireless environment. In addition, the result is frequently shown in a large screen format, not at all suitable for the smaller displays of the wireless devices.

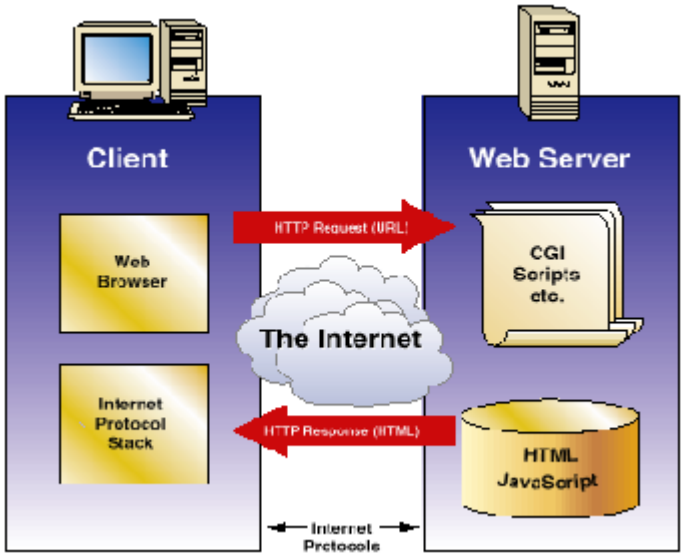


Figure 2.3: The Internet Model

In the WAP model the Internet standards have been optimized to match the wireless environment. Figure 2.4 shows this resemblance. The Wireless Markup Language (WML) and the Wireless Session Protocol (WSP) are mostly byte coded; that is, information packages sent are compressed, resulting in less load on the network. The counterpart format of HTML is the WML format, which is based on the eXtensible Markup Language (XML).

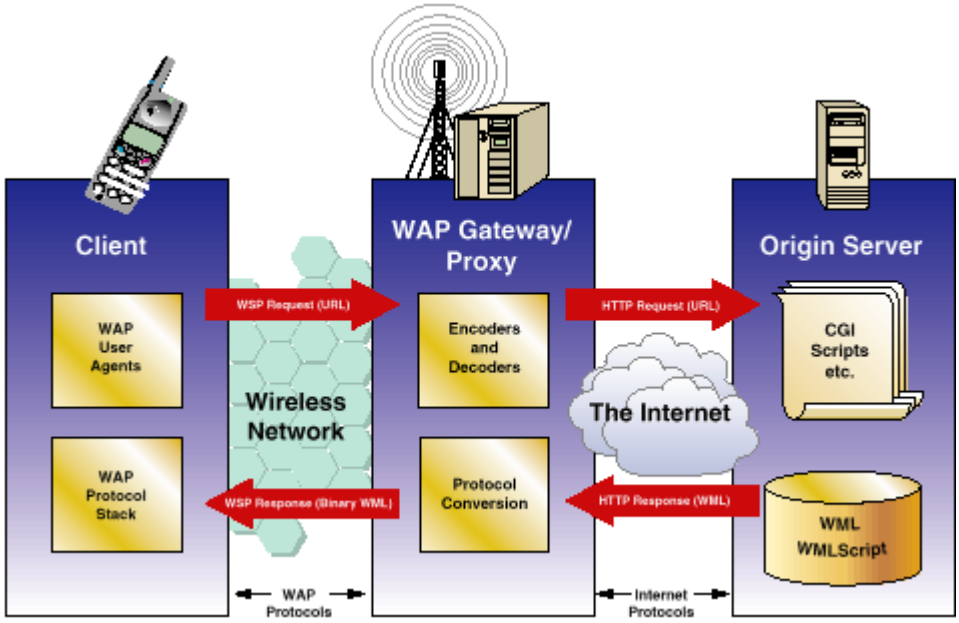


Figure 2.4: The WAP Model

The size of a display on a wireless device is always limited, which means that a large flat format (like the one used in HTML) cannot be used. Instead WML uses a deck/card metaphor to implement a service, as illustrated in Figure 2.5. When downloading a service to, for example, a mobile phone, a whole deck is received and the subscriber can browse between the cards locally on the phone. In practice this means that the server does not need to be contacted as often, and as a consequence the load on the wireless network decreases.

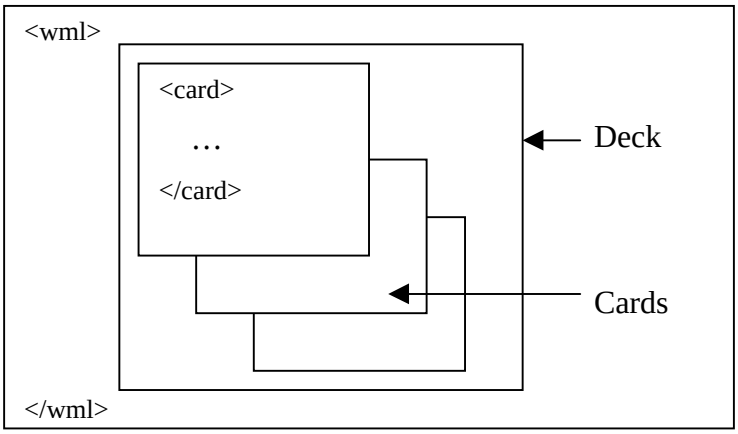


Figure 2.5: The Deck/Card Metaphor

The encoding of WML into byte code is one of the tasks performed by a WAP Gateway/Proxy, which connects the wireless domain to the Internet. Without a WAP Gateway/Proxy, the WAP model would have been almost identical to the Internet model.

Another difference is in the addressing model, where WAP uses not only URLs, but also other applications of Uniform Resource Identifiers (URIs) in order to access the locally based telephony functions of the wireless devices.

2.5 The WAP Architecture

The WAP architecture (see Figure 2.6) is founded on the principles of the Open System Interconnection (OSI) model, the five most important layers being:

- **Application Layer**

  The Wireless Application Environment (WAE) contains components for the use of Web-based services from a micro-browser as well as telephony services. The micro-browser services are built upon the addressing model, WML, and WMLScript. WMLScript is WAP's way of introducing procedural logic, loops etc. and is similar to JavaScript. WAE also provides a way to implement telephony services in the form of the Wireless Telephony Application (WTA) environment. WTA includes, among other things, the possibility of indicating the latest updates of Web sites by using the push mechanism. Push means that the server automatically sends new information to the clients that have requested the service, e.g. email notification. Another important service included in the WAE is the User Agent Profile (UA-Prof). UA-Prof is the component holding information about the capabilities of devices, and the user's preferences. This information is useful for the Origin Server or the Proxy/Gateway when they select the correct information to send, since it allows the subscriber to receive an adapted service suitable to his/her needs.

- **Session Layer**

  The Wireless Session Protocol (WSP) handles the connection to establish the session between a client and a WAP Gateway/Proxy. WSP is closely related to HTTP and could in fact be regarded as a binary version of HTTP, with the reservation that, unlike HTTP, WSP can be either connectionless or connection-oriented. WSP supports long-lived sessions, meaning that the connection can be suspended when communication is

temporary on hold, and later resumed. In other words, the user does not have to disconnect.

- **Transaction Layer**

  The Wireless Transaction Protocol (WTP) is responsible for the transmission of messages, and provides a reliable communication path. WTP is only used for connection-oriented requests.

- **Security Layer**

  The Wireless Transport Layer Security (WTLS) handles transport layer security between the WAP client and the WAP Gateway/Proxy. Privacy is guaranteed, meaning that it is possible to make sure that no one has tampered with a sent message by the means of encryption and authentication. This layer is optional.

- **Transport Layer**

  The Wireless Datagram Protocol (WDP) is used when the wireless network does not support the User Datagram Protocol (UDP). In reality this means that WAP uses a datagram service no matter if the network supports it or not.

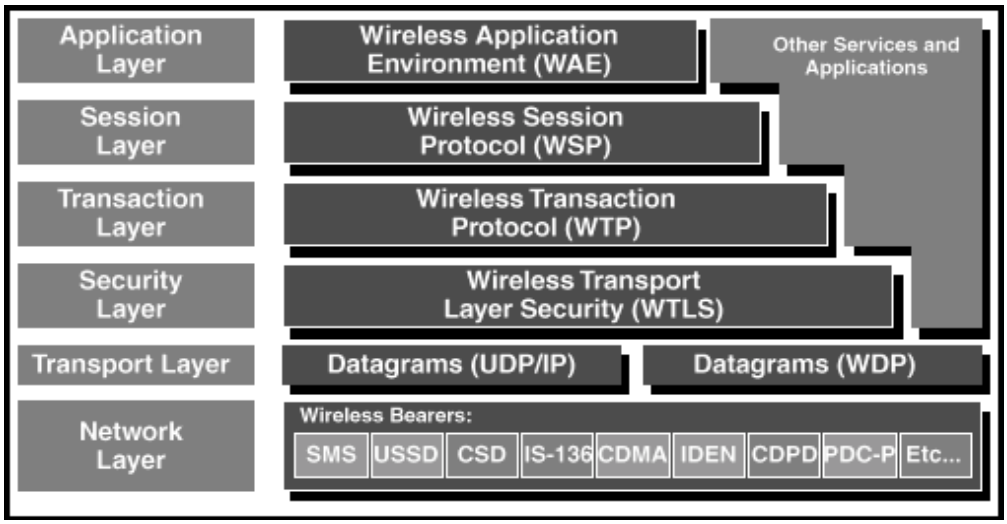


Figure 2.6: The WAP Protocol Stack

3 The Concept of Traffic Models

In this section we will briefly explain why traffic models are created, what information the traffic models provide developers with, and different ways to obtain useful information to base the traffic models on. The concept of traffic models is vague, and there are no defined guidelines in developing them. Yet they are of indisputable use for performance evaluation and capacity planning of servers, proxies and whole networks. Further information on traffic models can be found in [1, 4, 5, 6, 7, 8, 9, 10, 13, 14, 15, 16, 19, 21, 24, 26].

3.1 The Need for Traffic Models

Web traffic increases daily on the Internet, and it can be assumed that this will also be the case for WAP traffic when its services reach the every day user of the Web. An increase in traffic makes it important for developers to have good and precise traffic models in order to foresee possible traffic patterns. Exact traffic models of a system (e.g. network) and its workload is an essential tool for making realistic simulations. By being able to understand the impact of a systems workload, performance improvements can be made. A traffic model is therefore an asset in evaluation and engineering of communication networks.

3.2 Empirical Traffic Models

A traffic model is founded upon a statistical analysis based on information gathered about the system in question. What the traffic model should describe depends on which aspects are of interest to model in the system. Examples of some aspects that can be of interest to model are; traffic intensity, possible bottlenecks in the system and frequently requested information.

An empirical traffic model is founded upon the gathering of relevant information, such as client logs, server logs and packet traces, that characterizes the behavior of the system. Or, to put it in a different way, an empirical model can be developed using probability distributions determined by analysis of actual conversations. The aim of empirical models is to build communication patterns that show the characteristics of a network. The result should, however, be compared to other existing measurements and analysis, if such are present, to see if they are consistent. The different ways of collecting the information needed for the creation of an empirical traffic model are described in [19] and presented and summarized below:

- **Client logs**

  A client (e.g. browser) can log all retrievals made during a user session. One advantage with client logs is that they capture accesses between multiple Web servers. However, few browsers have the needed capability.

- **Server logs**

  The majority of existing Web servers keep logs on the requests they have served. These logs can be used to create a model of the workload. A disadvantage of this approach is that server logs cannot easily capture user access patterns across multiple Web servers.

- **Packet traces**

  A packet trace is collected from a junction in the network, for instance a router or a gateway, thereby eliminating the drawbacks presented in the methods above. It captures the behavior of individual users but misses higher-level information, such as specific files requested, and document types.

3.3 Workload Generation

An important aspect in development and performance evaluations of communication networks is the ability to perform simulations of the network workload. A simulation mimics the real behavior of a network, enabling decisions about, for instance, network structure to be made based on simulation results. A workload simulation of a communication network can be viewed as a way of generating a stream of requests reflecting the user behavior of the communication network. To be able to accomplish such a simulation two main approaches can be applied; a simulation based on static prerecorded user traffic, or a simulation based on a traffic model describing the behavior of the system and its traffic. These two basic ways to generate a synthetic workload are, as described in [4]:

- **Trace based workload generation**

  The trace based workload generation method relies on records of workloads that have been recorded in the past. To generate workloads, traces can either be sampled or replayed. The method is easy to implement, and imitates the behaviour of a known system, but it is hard to adjust the workload to varying demands and to mimic feasible

future conditions. Worth clarifying is that since this method samples or replays traces, it is not based on a traffic model and therefore it is hard to predict possible future behavior of the system if its prerequisites were to change. Trace based workload generation depends heavily on static information about the system, while a method based on a traffic model is more general and changeable.

- **Analytic workload generation**

  Briefly explained, the analytic workload generation method uses mathematical models (i.e. traffic models) for different workload characteristics, and then generates workload outputs that resemble the models. In this method it is important to identify the characteristics of the chosen workload, or workloads, to model. These characteristics must be empirically measured in order to yield reliable results. The main disadvantage of this method is the difficulty to create a single output workload that accurately exhibits a large number of different characteristics.

4 Statistical Distributions

This section contains a summary of the various statistical distributions used in this paper to compare to the given logged data. The purpose of this chapter is to give the reader a basic knowledge of some common distributions. We will briefly explain the Normal Distribution, the Lognormal Distribution and the Weibull Distribution. For further definitions of the distributions we refer to Appendix B.

- **The Normal Distribution**

  The Normal Distribution is one of the most common of all known distributions. The curve of the Normal Distribution is often resembled with a bell, because the majority of the measured data are collected in the center of the curve. The distribution can contain both positive and negative values, which means that in some measurements, it is not applicable. An example of a typical Normal Distribution is shown in Figure 4.1.

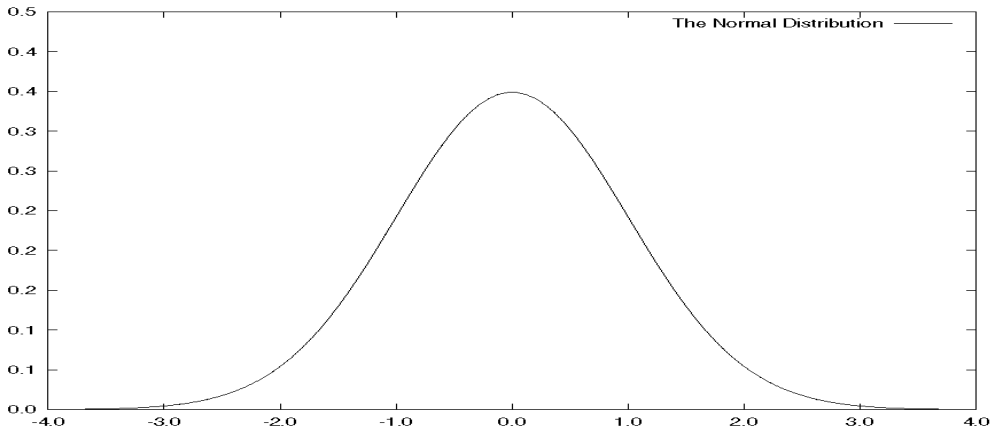


Figure 4.1: The Normal Distribution

- **The Lognormal Distribution**

  The Lognormal Distribution is actually a family of distributions, since its appearance can be skewed. Ordinary distributions, on the other hand, tend to have one curve with varying parameters. The Lognormal Distribution contains only positive values and its bulge is always situated at the beginning of the x-axis, that is, the curve is displaced to the left side

of the symmetric point. The distribution is an alternative to the Normal Distribution. An example of a possible curve is given in Figure 4.2.

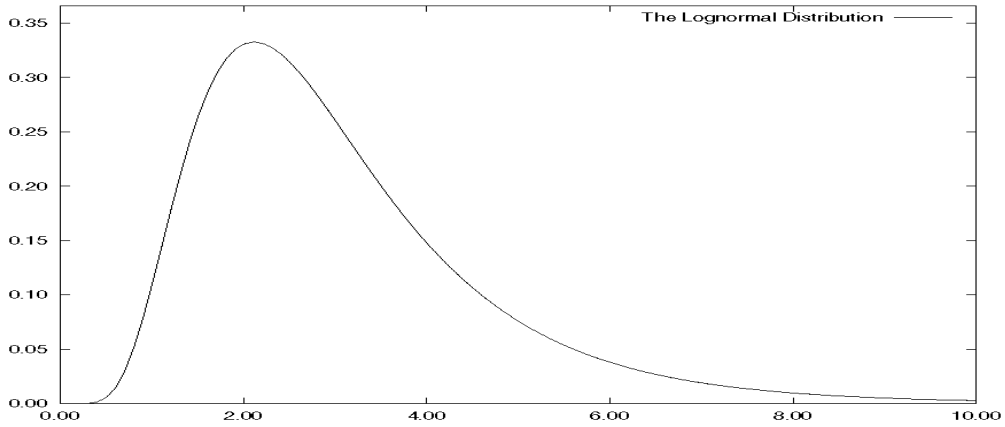


Figure 4.2: The Lognormal Distribution

- **The Weibull Distribution**

  This distribution can be used to describe the lifetime of an object, for instance, disc drives and light bulbs. The Weibull Distribution is sometimes a better alternative than the Normal Distribution, because the Normal Distribution allows negative observations. The shape of a typical Weibull Distribution is shown in Figure 4.3.

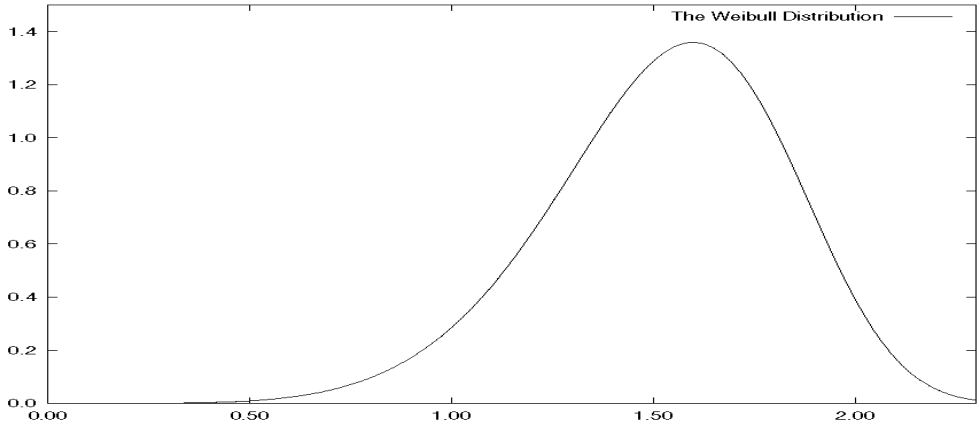


Figure 4.3: The Weibull Distribution

5 Measurements of WAP Traffic

This section presents the course of action taken when performing measurement analysis. We will explain what has been measured in order to create our traffic model, as well as how and why the measurements were taken.

5.1 Data Collection

The WAP traffic analysis in this paper is based upon data collected from a public WAP Gateway/Proxy (WGP) at Ericsson Software Technology AB in Karlskrona. The WAP Gateway/Proxy is a gathering point for WAP traffic since it receives requests from WAP clients, and then forwards them to Web servers as HTTP requests, as shown in Figure 5.1. The form of measurement performed at the WGP is based on packet traces (see Section 3.2) since it captures the behavior of many WAP clients.

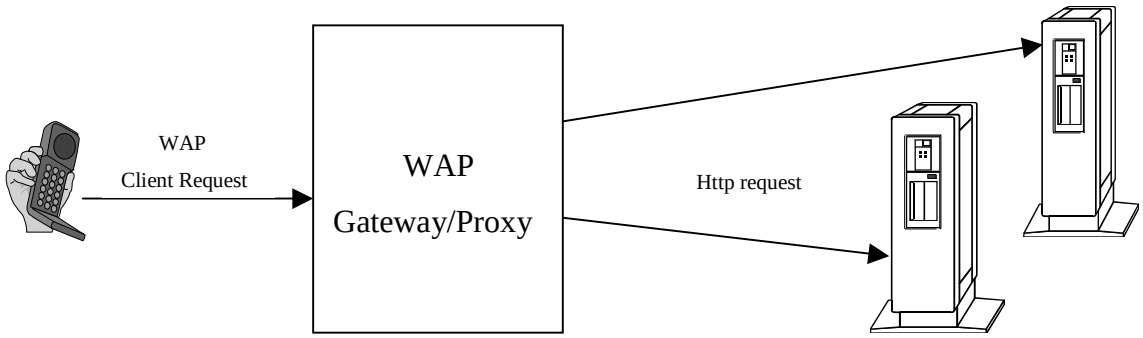


Figure 5.1: Functionality of the WAP Gateway/Proxy

Although the WGP is the best location to collect realistic measurements it also has some disadvantages, such as loss of specific information about file sizes and request types. However, as the advantages are larger than the disadvantages, we have chosen to accept this.

As explained in Section 2.5, WSP is a stateful protocol, which allows the WAP user to establish a session against the WAP Gateway/Proxy. A session could be compared to an ordinary telephone call, with the exception that WAP enables a user to suspend and resume an established connection, meaning that a session is open until explicitly removed. Because of this it is possible to measure the number of sessions established at a WGP. This measurement presents an image of user intensity during, for example, one day. The sessions generate HTTP requests towards various Web servers, making it feasible to determine a possible link between sessions and request intensity. For developers, knowledge about this link represents an

important asset in the evaluation and deployment of a system. Even though there are many interesting aspects worth modeling in the WAP area, we have in this paper chosen to focus on the correlation mentioned above.

5.1.1 The WAP Gateway/Proxy Logs

The logs, which have been used when developing our traffic model, have a certain syntax. Two examples are shown below in Figure 5.2:

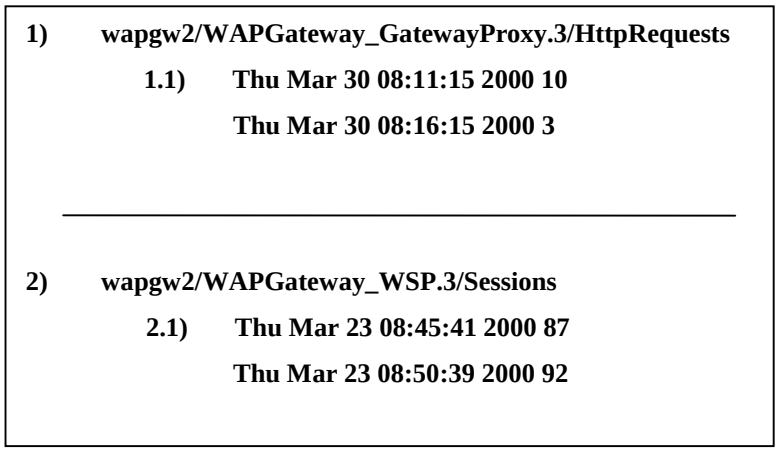


Figure 5.2: Examples of Log Entries

The WGP uses a five-minute sampling time, meaning that every fifth minute it reads and logs the value of a specified counter. To give the reader a further insight into the log entries we will here provide a short description of the varying parameters. A general log entry is of the form according to 1) and 2) in the figure above:

<WGP name>/<WGP connection mode>/<specified log counter>

The first parameter, *WGP name*, in the example above specifies the name of the WAP Gateway/Proxy. The WAP Gateway/Proxy can log both connection oriented and connectionless counters. At the end of the *WGP connection mode*, the number three stands for a connection oriented mode, and the number four stands for a connectionless mode. The last parameter, *specified log counter*, gives the occurrences measured, which could be the number of HTTP requests or the number of sessions established.

In Figure 5.2 the sub point 1.1) represents a log for HTTP requests according to the following syntax:

<Weekday> <Month> <Day> <Time for log entry> <Year> <Number of HTTP requests>

The parameter *Number of HTTP requests* is the total number of requests during the five minute sampling time. The other parameters are self-explanatory, especially when compared to Figure 5.2.

In Figure 5.2 the sub point 2.1) represents a log for the number of sessions according to the following syntax:

<Weekday> <Month> <Day> <Time for log entry> <Year> <Number of Sessions>

The parameter *Number of Sessions* is the number of sessions open at the specific log entry time. Worth mentioning is that the sampling time captures the number of open sessions or in other words, a session open during one sampling time can be a part of several more sampling intervals. Until a session is closed by the user or terminated by the WGP it will appear in the log entries. The WGP contains default parameters that automatically close sessions that have been inactive for a specific time.

5.2 Methodology

As a first step we tried to learn as much as possible about the concept of traffic models. This step included a thorough search of the Internet since the Web turned out to be our primary source of information. We concentrated on Internet computer science bibliographies and from papers that we found there we could continue our investigation. The papers also contained references to other similar reports on the subject (i.e. traffic models) that proved to be useful. A more general insight into traffic models was difficult to obtain; information presented in this report has been gathered from traffic models describing other areas, such as HTTP.

The next step was to decide from where the data was to be collected. A few alternatives existed but most of them were unobtainable and some would not yield the generality we preferred. The gist of our alternatives was one acceptable source of data, namely a WAP Gateway/Proxy. At this point our quest for an available WGP began.

The third step was achieved when we finally contacted Ericsson Software Technology AB in Karlskrona. They were willing to contribute with logged data from their public WGP. It was now up to us to filter the logs for what we considered useful information. For this purpose we created four JAVA programs: LogParser, LinkParser, AverageParser and RequestParser. LogParser filters out the time and the number of HTTP requests or

alternatively the number of sessions. It also gives the total amount of requests/sessions and the total number of entries. RequestParser takes two filenames containing HTTP requests from the two WGP connection modes (see Subsection 5.1.1), and adds the number of HTTP requests from these two modes, producing a new log file. LinkParser also takes two filenames, the first containing HTTP requests (filtered by RequestParser) and the second sessions (filtered by LogParser) and creates a new file containing the link between sessions and HTTP requests with respect to time. AverageParser takes two files, the first containing HTTP requests and the second sessions, one filtered by LogParser and one filtered by RequestParser, and computes the average number of HTTP requests per session and per sampling time. The programs are presented in Appendix C. Table 5.1 summarizes the characteristics of the gathered data used in this report. The contents of the table are for the larger part self-explanatory. There are, however, a few exceptions; the dates for the logged data stretches from Mars to April but this report is only based on twelve (non consecutive) days in this time interval, as the log duration specifies. In other words, we have had access to measured data from twelve days, where the dates range over one month. Another detail worth clarifying is the maximum number of sessions, which indicates the maximum number of sessions established at any given time during the logged period.

Log Duration	12 days
Log Start Date – Log End Date	03-20-2000 – 04-20-2000
Total HTTP Requests	138 001
Average HTTP Requests / Day	11 500
Maximum Number of Sessions	230

Table 5.1: WGP Log Characteristics

Step number four was to analyze the filtered out data. To achieve the best basic conditions we first had to create a graphical representation of the logged data. This was accomplished through the use of the program GnuPlot, with a great deal of help from the useful GnuPlot User's Guide [17]. When comparing the plotted results we tried to spot a direct resemblance with common statistical distributions (see Section 4), such as The Weibull Distribution.

The final step was to analyze the reasons why our result had a certain appearance. When possible, oddities in our plotted data were examined for feasible background explanations. The results of our analysis are presented in the next section.

6 Modeling Results

In this section we will present our modeling results. By introducing some samples of curves generated from the filtered logs, we hope to give the reader an insight into the current WAP traffic situation. The graphs of statistics shown in this section have been chosen because they represent some interesting or general behavior of the WAP traffic modeled in this report. Most of the graphs presented will be based on seven or eight explicitly chosen days, because these days contain an almost complete set of data. In our analysis we will attempt to highlight and to a certain extent explain the modeled results. The interested reader can find graphs for all measured days in Appendix D.

Since it is of great importance that the reader understands the purpose of this modeling analysis, we feel that a brief repetition of our choice of modeling components is in order. As mentioned earlier, this report is aimed towards modeling the feasible link[1] between the WAP users, and the WAP traffic they generate. The nearest correlation we were able to make out resembling that possible link was the relationship between the amount of sessions established, and the HTTP traffic they bring forth. The underlying assumption being that one session would map to one WAP user.

6.1 A Background to Our Model Analysis

Before creating our graphs we discussed which parameters could be of interest to consider in our model. Obvious parameters were sessions and HTTP requests because they represented the basic assignment for this report. The day and the time of the day were other plausible parameters of importance. The line of reasoning went as follows; certain weekdays could have more traffic than other weekdays and a day could contain peek hours. The same reasoning can be applied to specific hours during a day, where peek hours would represent an increasing production. We feel it is plausible that a peek hour could be represented by the increasing production of HTTP requests per session, or simply by an increase of the number of sessions. On the other hand, there is also a probability that a relationship between WAP traffic and time does not exist.

[1] The observant reader may have noticed our frequent use of “a possible link between users and the traffic they generate”. This is done intentionally to remind the reader that this report aims to determine if such a link exists.

6.2 Modeling Overview

Before we had access to the observed data we had an idea about being able to fit the data to existing probability distributions. If this approach had been applicable it would have given a more general, and easily changeable model of the WAP traffic. Unfortunately, such a curve fitting approach could not be performed because nearly no similarities to any probability distributions we used for comparison (see Section 4) were found. Instead we were forced to find alternative ways to investigate any patterns in the plotted data. We tried to divide and focus on possible patterns in a smaller time interval. A few graphs were produced to observe possible patterns in the session periods. These session periods resemble “shark teeth” and are easily be spotted in Figure 6.1. However, this attempt turned out to be futile, since no patterns emerged, and therefore none of those graphs have been included in this paper.

As a step to see possible patterns in the received logs, we plotted the number of sessions established, and the HTTP requests they generated during a day, in two different graphs. In the next step we plotted the sessions and HTTP requests together in one graph to make a feasible pattern even clearer, see Figure 6.1. These two steps were taken to get a first glance of the WAP traffic characteristics we were modeling. Since we tried to model the correlation between sessions and HTTP requests we then made an attempt to observe a pattern where the amount of sessions and their HTTP requests did not stand in relation to the time of day. We were nevertheless still unsure of the significance of the time factor, so we decided to include it in the correlation between sessions and HTTP requests, producing three-dimensional graphs. In the following section we will present our observations of the WAP traffic behaviour.

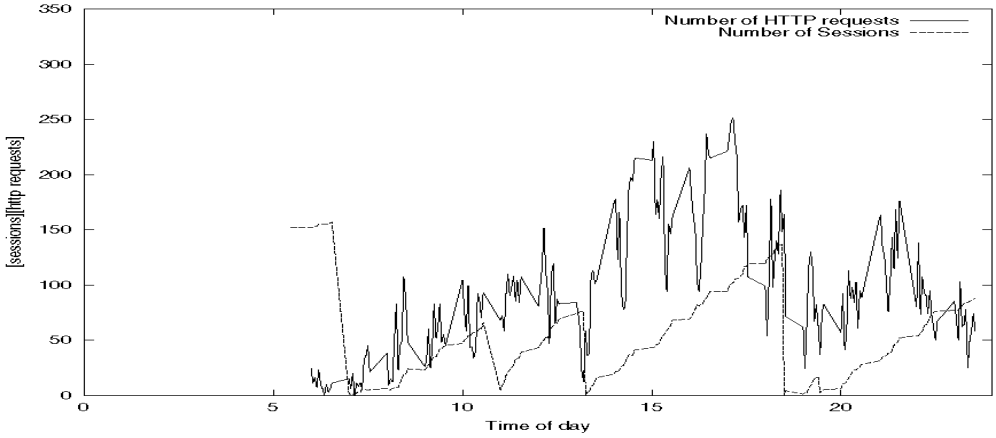
6.3 Model Analysis

Different aspects of session and HTTP request behavior will be explored in this section. To elucidate these aspects we have chosen to present our model analysis in a specific way. First, we give some example graphs, and then we discuss, and attempt to explain the appearance of them. We hope this arrangement will allow the reader to achieve a deeper understanding of our results.

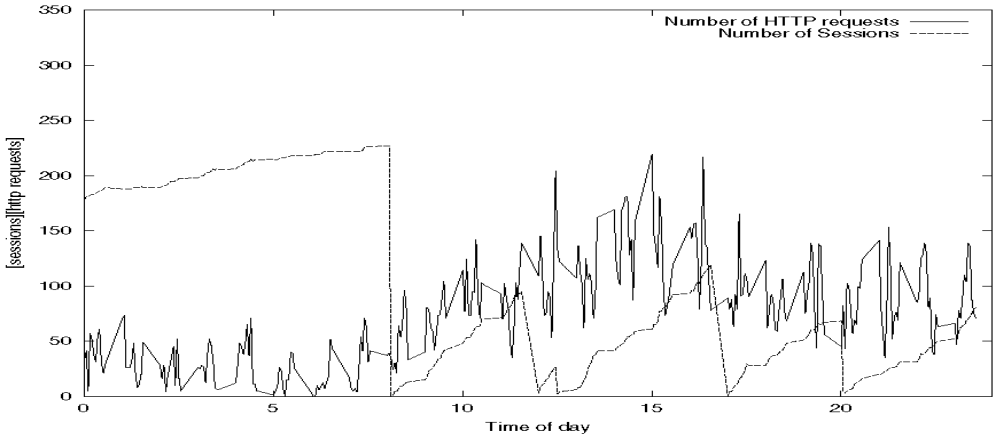
6.3.1 The Traffic Intensity during a Day

The starting point of our analysis is the study of traffic intensity during one day. The graphs in Figure 6.1 show the number of sessions and the HTTP requests they produce. The somewhat

cryptic appearance of the sessions can to some degree be explained by the fact that sessions that have been passive for a certain time will be removed according to predefined time interval parameters in the WGP. According to the graphs, the number of active sessions established at the WGP seems to grow in a linear style. Unfortunately, there are no proof supporting this phenomenon, since there was no way of determining the number of passive sessions in the graphs. Therefore it was difficult to draw any reliable conclusions concerning the assumed correlation between sessions and HTTP requests.



(a) Observation for 03-20-2000



(b) Observation for 03-30-2000

Figure 6.1: Observed Sessions and HTTP Requests

If the varying deviations in the HTTP requests are disregarded when analyzing the plotted data shown in Figure 6.1, similar patterns can be made out. During the first hours of the day, the numbers of sessions are high but they do not produce particularly much HTTP request traffic. We assume this scenario can be partly explained by the fact that the WAP Gateway/Proxy does not remove passive sessions during the night. That would indicate the

existence of only a few active sessions, resulting in fewer HTTP requests than during the rest of the day. Logic would support this theory, since most people tend to sleep during these hours. As for the rest of the day, the HTTP requests resemble the shape of a flat Normal/Lognormal Distribution, reaching its top values around 15 hours, suggesting a peak/busy hour in HTTP request traffic intensity.

6.3.2 The Correlation between Sessions and HTTP Requests

This subsection came into existence when we tried to model the correlation between sessions and HTTP requests.

Because of the difficulties in modeling the number of active sessions in our measured dataset, we have made the assumption that *all* sessions are active until the point when they are physically removed. It cannot be enough emphasized that most probably, the graphs, tables and observations presented below are not representative for WAP traffic in general. This is due to the fact that the linear growth of sessions that abruptly dives to almost zero (as shown in Figure 6.1) would indicate an increasing number of passive sessions all removed according to the defined parameters in the WGP. Or more concise, all sessions cannot be active since they are almost all removed simultaneously; therefore we may here present a false image of the number of HTTP requests produced by the real active sessions.

We will refer to the HTTP requests produced by a session as the *request rate*. This is done to further improve readability. The request rate can be calculated over different time intervals, for example during the five minute sampling time or over one or more days. The time interval used will be made clear by the context.

Figure 6.2 gives a picture of the request rate, during the five-minute sampling time, observed for a period of seven explicitly chosen days, see the table in Appendix E.1. To obtain a complete picture of the HTTP request traffic at the WGP, we have considered the combined HTTP request traffic produced by the two connection modes mentioned in Section 5.1.1, meaning that the traffic produced by the different modes have been added. To reach the results shown in Figure 6.2 we have for each sampling time entry divided the number of HTTP requests with the number of open sessions.

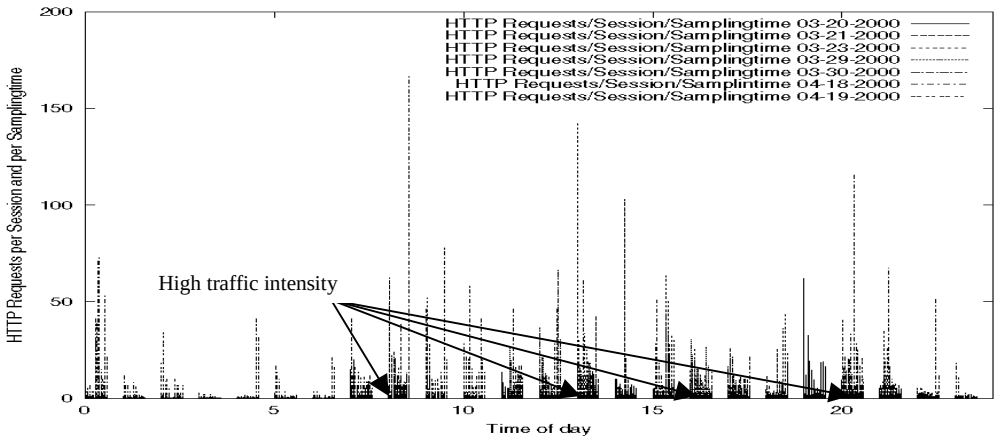


Figure 6.2: Request Rates during one Day

The figure shows where the request rates are situated during one day. An observation that can be drawn from Figure 6.2 is for instance that the darker parts plotted, highlighted by arrows in the figure, represent the common pattern for sessions and the HTTP requests they produce during one day. Except for some fluctuations in the plotted data, the intensity seems to be concentrated around four time periods; 7 hours to 9 hours, 11 hours to 14 hours, 15 hours to 18 hours and finally 19 hours to 22 hours, each period forming small versions of Normal Distributions. The intensity appears to be related to a "normal" working day at any Ericsson branch[2]. The figure applies detailed information of the lower portion of the traffic as well as it illustrates the upper portion of the frequencies for peak rates. Traffic is low during morning hours and then increases to an even level with about five to ten requests per session, with occasional increasing deviations in the time periods mentioned earlier.

Worth to mention, in relation to Figure 6.2, is the fact that the time periods per day used to create the graph in the figure does not always have the same range. In other words, the plotted result could perhaps have been of a somewhat different appearance if all days had had the same time interval.

The next point worth considering when dealing with the behavior of sessions and HTTP requests would be the request rates calculated over a day and the request rate calculated over our entire measurement period. The results presented in Table 6.1 and Table 6.2 have been reached by dividing the total number of HTTP requests produced throughout the specified time interval with the total number of sessions measured. The total number of sessions

[2] The observed data has after all been gathered from an open WGP, mostly used by Ericsson employees.

include duplicated sessions as explained in Section 5.1.1. To make this plausible, the users would have to disconnect their established sessions for each sampling time, a behaviour not suggested by Figure 6.1 where few users seem to disconnect their own sessions.

Date	**Request Rates Calculated over a Day**
03-20-2000	1.688817
03-21-2000	0.916303
03-22-2000	1.770466
03-23-2000	2.523139
03-24-2000	1.506992
03-29-2000	1.367706
03-30-2000	0.739582
03-31-2000	0.312269
04-17-2000	6.585580
04-18-2000	4.256595
04-19-2000	1.534953
04-20-2000	2.198140

Table 6.1: Request Rates Calculated over a Day

In table 6.1, the request rates presented vary substantially. In part this can be explained by the fact that we have had different amounts of logged data (see Appendix E.3) to base our calculations on, as we have already mentioned before in this section. Another important aspect worth considering in relation to the results in the table above is the behaviour of the sessions in correlation to the number of HTTP requests they produce, and when they produce them. For instance, during the night a high number of sessions produce a relatively low number of HTTP requests (see Appendix D.1 for examples), affecting the resulting request rates negatively. In some cases, the sessions do not even produce one HTTP request. When calculating the request rate over the twelve measured days, we reached a request rate of approximately one according to the table presented below.

Request Rate Calculated over all 12 Days
1.353004

Table 6.2: Request Rate Calculated over 12 Days

If more detailed measurements of active and passive sessions could have been performed we feel that the graphs and calculations made in this subsection would have given a more accurate result. As for the line of reasoning in this subsection we ask the reader to take the presented observations lightly.

To further investigate the correlation between sessions and HTTP requests we will start by introducing a graph, Figure 6.3, showing the link between the number of sessions and the HTTP requests they produce. Each point in the graph represent the number of HTTP requests produced by a given number of sessions during one sampling time. To begin with, we have disregarded the time and day as interesting parameters in an attempt to find general characteristics in a feasible correlation. The graph in Figure 6.3 is a union of all whole sets of logged data in an attempt to enhance the possible existence of such a correlation. Then we reintroduced the time interval in three-dimensional graphs (see Figure 6.4) to investigate all possible aspects. To achieve a more dependable view of the traffic distribution, the graphs in Figure 6.3 and 6.4 are based on data from the table in Appendix E.1; i.e. the days containing the highest frequency of logged data. The different days are represented by various labels.

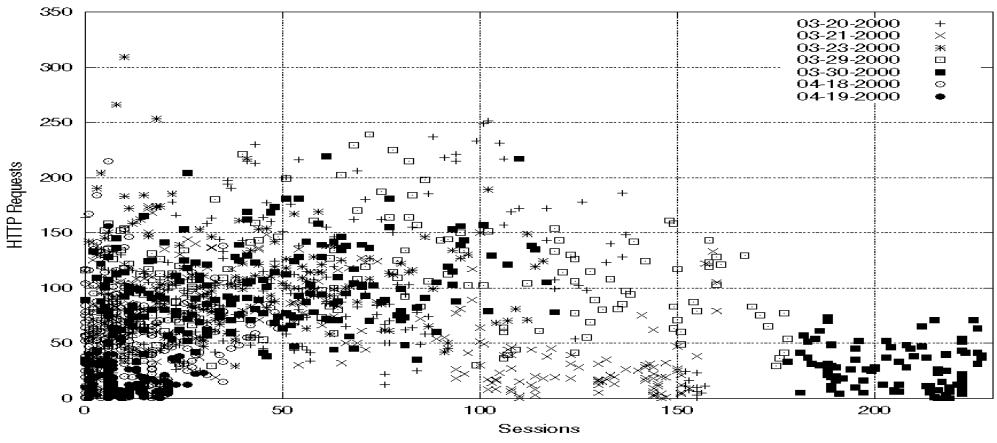
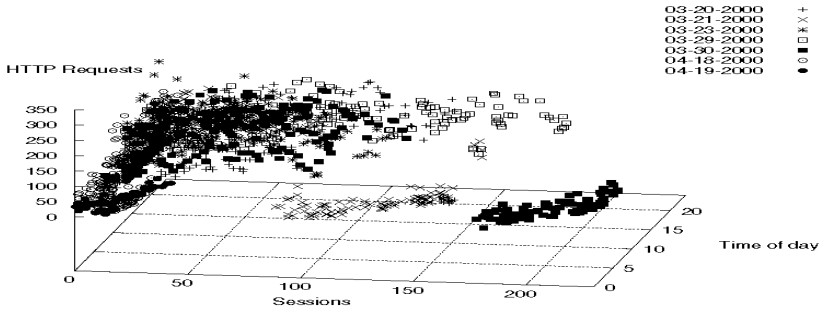


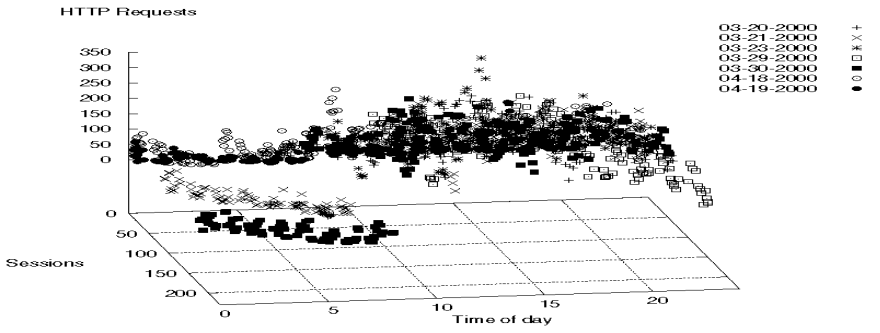
Figure 6.3: The Correlation between Sessions and HTTP Requests

Several observations can be made from the graph above; the largest accumulation of sessions/HTTP request points are gathered approximately in the area 0-40 sessions producing

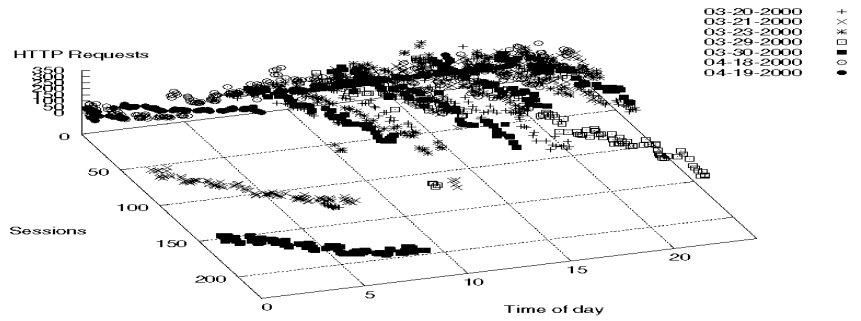
up to 150 HTTP requests, then the data points are more sparsely situated. Another observation that could be made from the graph in Figure 6.3, is the fact that no number of sessions produce more than 350 HTTP requests, and, as the number of sessions increase, the less HTTP requests they seem to produce. The latter could perhaps be explained in retrospect to the line of reasoning in Subsection 6.3.1, when we presented Figure 6.1. We then introduced the assumption that during the night a high number of sessions produced a low number of HTTP requests. As no certain conclusions, such as for example a linear growth, can be drawn from the graph in Figure 6.3 we feel that the time factor may be of importance. To verify our assumptions we have chosen to include the time parameter in our graphs, thereby producing three-dimensional plots (see Figure 6.4).



(a) Focus on Sessions and HTTP Requests



(b) Focus on HTTP Requests and Time



(c) An Overall View

Figure 6.4: The Correlation between Sessions and HTTP Requests over Time

The graphs shown in Figure 6.4 all highlight different aspects of the line of reasoning stated earlier in this subsection. Figure 6.4 (a) illustrates that the number of active sessions normally is in the interval 0-50, Figure 6.4 (b) illustrates that those sessions are most often situated in the time interval 12 hours to 23 hours with the center around 18 hours, producing up to 150 HTTP requests. Figure 6.4 (c) verifies our assumptions in an overall view.

From Figure 6.4 we can once again confirm our theory that during the night and morning hours few passive sessions are removed from the WAP Gateway/Proxy, leaving only a few "real active" sessions to produce the recorded HTTP Request traffic.

As for any certain conclusions, we see none that can be statistically applicable. However, we believe that the time parameter should be of significance and that most of the graphs shown in Section 6 supports this belief.

6.3.3 The Correlation between Sessions and HTTP Requests from Another Viewpoint

The results from Subsection 6.3.2 were somewhat misleading, and therefore we will here attack the correlation between sessions and HTTP requests from another viewpoint.

The number of sessions in Table 6.3 is the closest we can come to the number of users during one day. Due to the unique appearance of the graphs (see Figure 6.1 for examples), suggesting that all sessions are passive when removed according to specific time periods, it is feasible to assume that users mostly do not remove their sessions after usage. To reach the stated number of sessions during a day, as given in Table 6.3, we had to make the assumption that all the maximum session apexes measured during one day could be used as an indication of how many sessions were established during the measured time interval. We feel this is a somewhat uncertain estimation, but that it is more accurate than the estimation made in the previous subsection.

We have mainly concentrated on displaying our observed data in a graphical format, however, it may be useful to examine the results in a table format. Especially since we have removed the time aspect from our calculations. Table 6.3 contains data concerning; measured time intervals during a specific day; the number of opened sessions during that day; and the total amount of HTTP requests produced.

Date	**Time Interval**	**Sessions**	**HTTP Requests**
03-20-2000	06.00-23.55	410	20 357
03-21-2000	00.00-07.40 12.00-23.55	467	15 546
03-23-2000	09.00-23.55	433	18 810
03-29-2000	11.01-23.56	475	17 017
03-30-2000	00.01-23.56	439	21 138
04-18-2000	00.03-23.57	374	15 305
04-19-2000	00.00-07.57 09.20-23.56	62	3 096

Table 6.3: An Overview of the Measured Traffic Situation

The number of sessions mostly varies between 374 and 475, as stated in the table above. Because of this we feel that the measured data from 04-19-2000 can be seen as somewhat of an anomaly. A more thorough investigation showed 04-19-2000 to be the day before Maundy Thursday, a day where employees at Ericsson are likely to be off work.

Based on the new estimation of the number of active sessions during a day we can now calculate the request rate in an alternative fashion. From the data in Table 6.3 we can obtain the request rate for each of the seven days by dividing the number of HTTP requests with the estimated number of opened sessions. The results are presented in Table 6.4.

Date	Request Rates Calculated over a Day
03-20-2000	49.651220
03-21-2000	33.289079
03-23-2000	43.441109
03-29-2000	35.825263
03-30-2000	48.150342
04-18-2000	40.922460
04-19-2000	49.935484

Table 6.4: Request Rates Calculated over a Day

Furthermore, the request rate over the seven measured days is presented in the table below. This result has been reached by dividing the total number of HTTP requests produced with the total number of sessions; both measured over the seven-day period presented in Table 6.3.

Request Rate Calculated over Seven Days
41.830451

Table 6.5: Request Rate Calculated over Seven Days

Using the assumption made in this subsection we feel that most of the duplicated sessions that occurred in the graphs and calculations of Subsection 6.3.2 have been successfully disregarded, producing more reliable results as a consequence. From the tables above we can see that the calculated request rate is now around 42. From Table 6.4 we can also draw the conclusion that the individuals making use of the WGP on 04-19-2000 produced about as much traffic as they would during a “normal” day.

6.3.4 Summarization of Results and Discussion of Future Traffic Behaviour

So far we have discussed and observed certain patterns based on our measured dataset in an attempt to see if any characteristics can be found. In this subsection we will try to summarize our observations, and give a brief discussion of possible traffic patterns in the future. Some of the discussion will be in reference to general WAP traffic behaviour (if so specified), but the larger part will be in reference to our measured dataset.

We have made two different assumptions during our evaluation of WAP traffic. In Section 6.3.2 we assumed that all sessions were active, which as stated in that section, is not true. The

whole section presents a false image of the correlation between sessions and HTTP requests. The results could, at their best, present an indication as to the real correlation between the sessions and the HTTP requests they produce. The request rate in Section 6.3.2 should increase if the measurements were based on only real active sessions. In our case the results should improve if the data measured during the night are disregarded. We have seen that the request rate during the twelve-day period is approximately one (as shown in Table 6.2) in our dataset. When we tried to find a correlation between sessions and HTTP requests we found that it was hard to spot a direct common denominator. Adding the time parameter did not make this quest any easier. However, we were still able to observe that the number of active sessions in our measured dataset normally lies in the interval 0 to 50, and that the largest percentage of traffic seems to be situated between 12 hours and 23 hours, generating up to 150 HTTP requests (see Figure 6.4).

We also tried estimating the number of active sessions as the maximum number of sessions occurring in each session period. Our log measurements (seven days) have shown that between 374 and 475 sessions produces 15 305 to 21 138 HTTP requests, and that the request rate is approximately 42. We believe this to be a more accurate estimation.

A possible goal in future WAP traffic models may perhaps be to investigate if a possible connection between sessions and HTTP requests can be established and expressed by a mathematical formula, applying a number of sessions to generate the amount of HTTP requests the sessions most likely will produce. We believe that such a formula may be of great interest for WAP developers, especially when dimensioning equipment, for example WAP Gateway/Proxies. Unfortunately, the time factor and the quality of the measured dataset were too limited for us to reach any certain conclusions in this paper.

Possible peak hours in future WAP traffic may exist in the time intervals 7 to 9 hours, 11 to 14 hours and 16 to 20 hours. The first time interval could perhaps show the morning behavior of WAP users for instance when they are heading for work. The WAP usage during the lunch hours can exist in the second interval and the third time interval may represent the WAP usage during the evening, for example when the majority of the population has finished working. We have no proof that these peak hours will exist in future WAP traffic, we have simply given our ideas founded on well-known common sense, the fact that users of mobile phones seems to operate during these hours, and the few trends spotted in our analysis of log data.

7 Improvement Propositions

The purpose of this section is to emphasize some improvements we feel should be considered in possible future examinations of WAP traffic.

If we had had access to logs from existing WAP services on the market, such as those offered by Telia AB and Europolitan AB, a more reliable analysis of WAP user behavior could have been made. Unfortunately, they marketed their services rather too late for us to take them into consideration in this report. Attempts were made to contact Europolitan AB, but they where reluctant to participate in our investigation, because their logged data contained classified information. However, filtration of sensitive information should not be difficult to perform, and we suggest further contact concerning collaboration.

Another desirable improvement is an increased period of logged data, preferably from several sources. This should pave way for a more secure statistical base. The logged data should be prolonged to a minimum of one month. The larger the amount of data gathered, the more secure statistical conclusions can be drawn.

Furthermore, it would be more correct to measure only the active sessions at the WAP Gateway/Proxy if the correlation between sessions and HTTP requests should be investigated in depth. In our measurements all open sessions were recorded as active, even the passive ones, and that fact deteriorated our results in our first estimation since duplicated sessions were included in the calculations. We believe we reached a better result when we made our second estimation where we tried to estimate the total number of open sessions during one day. However, none of the estimations should be a problem if the WGP logs both active and suspended sessions.

We also propose an extended analysis of the WAP traffic, since other interesting correlations may exist. To give some examples, page sizes of requested services, the traffic (at packet level) on both sides of the WAP Gateway/Proxy and the traffic load on the WGP. Another aspect worth studying is user behavior at the wireless terminals, for example identifying which WAP services are most frequently used, and how.

We feel that a further evaluation of the WAP traffic is called for when WAP has become an accessible service to the greater part of the public. We believe that an evaluation at that stage would present a more accurate statistical base, and that the WAP traffic characteristics should be easier to spot. This report has taken the form of becoming a feasibility study, an aid

for future development of WAP traffic models, instead of explicitly developing a traffic model.

8 Alternative Solutions

In this section we will focus upon the different solutions that could perhaps be applied to capture relevant WAP traffic behavior. We will only present our ideas; this section does not give conclusive instructions on how they should be performed.

This report has taken the approach of modeling the WAP traffic at an intersection, i.e. at a WAP Gateway/Proxy, but this is, as mentioned in Section 3.2, not the only possible way. Another interesting factor worth concentrating on when modeling the WAP traffic could be to model the activity of the WAP entities, that is, the WAP devices. This could be done by making the WAP micro-browser log all of its activities. By tracing the behavior of a number of specifically chosen WAP users, a picture of their WAP usage should appear. This would be of importance when analyzing for example; file sizes of downloaded services; most frequently visited sites; and during what hours the normal WAP user utilizes the WAP services. The suggested approach would, however, intrude on the integrity of the WAP user, and would also demand specifically built micro-browsers that enable logging.

The WAP Gateway/Proxy approach and the WAP entity approach both models the overall picture of WAP user behavior, the WAP entity approach being a somewhat better but more complicated alternative. It would also be possible to model the Web servers that contain the WAP services. This approach would perhaps give the developer of the Web server a hint of the load on the server, but it would not present the comprehensive view offered by the two other approaches, and is therefore a more limited source of information.

9 Problems

This section is a way for us to inform the reader of some of the difficulties we have encountered during this assignment. We have no interest in making this chapter a section of complaints, but to make an effort to prevent the same mistakes from being made again.

9.1 Information Gathering

The first problem we encountered when searching for information on traffic models was the lack of general information concerning traffic models. All information conveyed in this report have been gathered from papers on existing traffic models.

Due to the limited availability of real traffic data concerning WAP traffic, researchers have not yet obtained reliable estimates regarding important key parameters in the WAP traffic. Consequently, little was done in the area of WAP traffic modeling when we started our project, and the existing papers were hard to come by and did not contribute with much information.

To be able to develop an empirical traffic model we had to found the traffic model on logs captured from a WAP Gateway/Proxy that described the WAP traffic in the real world. This log capturing seems very easy, but looks can be deceiving. Finally, we made contact with Ericsson Software Technology AB in Karlskrona which could provide us with logs from their WAP Gateway. This Gateway is available only to people that have knowledge of the IP-address (i.e. Ericsson employees), which means that the logs may not accurately reflect the real behavior of WAP users.

9.2 Traffic Model Development

In the beginning of this assignment problems arose concerning the requirement specifications. The requirements given did not match with the requirements expected according to our industrial supervisor. This resulted in a new formulation of demands, and it was somewhat time consuming to reevaluate our assignment, and adjust accordingly. Since the concept of traffic models was new to us, we often lost sight of the specified aim of this project. Also, to be able to accomplish a result within the given time, we soon realized that the original assignment had to be reduced. This reduction reflected itself in the development of our traffic

model. We decided, in agreement with our industrial supervisor, to focus upon one aspect of the WAP traffic, namely the proposed correlation between WSP sessions and the HTTP requests they produce. The next problem was, as stated in the section above, to be able to study the WAP traffic situation. When we finally received logged information, it contained several deficiencies, for instance, the logs did not cover one whole day. We partly solved this problem by downloading the logs ourselves. However, the largest disadvantage lay in the fact that we could not distinguish active sessions producing HTTP requests from the passive, abandoned sessions. This presented a false image of the WAP traffic situation.

10 Concluding Remarks

In this section we will present and summarize our final conclusions concerning the given assignment. We want to emphasize that this section will not touch upon the specific WAP traffic modeling results, that has been handled in Subsection 6.4.5. Instead this section will have a comprehensive starting point.

What the WAP Forum started when they introduced the Wireless Application Protocol, and when they were able to keep the standardization uniform, is nothing less than a wireless era without limitations, only possibilities. So far the WAP market is in its infancy, but the demands for WAP devices and services increase on an everyday basis. Since the demand today exceeds the range of WAP devices/services available it should be of great interest to investigate which demands the system may encounter.

The subject area is interesting, and the need for precise traffic models is obvious, both in development and evaluation of systems. We see both advantages and disadvantages in developing a WAP traffic model at this early stage. Amongst the advantages is the fact that a WAP traffic model showing the behavior of WAP users today, would be a useful preparation tool for further expansion. The main disadvantage is the fact that WAP is still in the developing phase and not widely spread/used, which means that captured traffic might not reflect future user behavior.

The making of a traffic model is not the easiest of tasks since there are no general rules or guidelines, as we discovered the hard way. To make a traffic model for Wireless Applications, a new field within data/telecommunications that has not yet been fully evaluated, is even more difficult. The WAP Gateway/Proxy used in this study is mainly utilized by Ericsson employees; some of them working actively with WAP devices during working hours. It is uncertain if their behavior would be reflected in the behavior of the mass population when the WAP services enter the public telecommunication market. Even though the need for a WAP traffic model is understandable a better and more secure result could perhaps be achieved when the WAP market has increased.

So far we have concluded that the topic of traffic models is interesting, and that they are strenuous to develop. Both of us accepted the challenge, and even though we met considerably more difficulties than expected, we feel that we have achieved a deeper understanding of the areas surrounding this assignment. However, we would have liked to be

able to make more general assumptions regarding the WAP traffic studied in this project. The evaluation became merely observations since we did not have enough proper statistical information to draw any certain conclusions. It is our sincerest hope that the results and procedures presented in this report will be of future use in an upcoming and further development of a WAP traffic model.

References

[1] Martin F. Arlitt and Carey L. Williamson. Web Server Workload Characterization: The Search for Invariants (Extended Version). Available via URL http://www.cs.usask.ca/projects/discus/discus_reports.html since May 1996.

[2] AU System Radio AB. Available via URL http://www.wapguide.com

[3] AU System. WAP White Paper. Available via URL http://www.wapguide.com/wapguide/wap.html since February 1999.

[4] Paul Barford and Mark Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Proceedings of the ACM SIGMETRICS Conference*, 1998.

[5] K. Chandra and A.E. Eckberg. Traffic Characteristics of On-Line Services. In *Proceedings of 2nd IEEE Symp. on Computers and Communications*, pp. 12-22, Alexandria, Egypt, July 1997.

[6] Lily Cheng and Herman D. Hughes. An ATM Traffic Model Based on Empirical Traffic Measurements. In *Proceedings of IEEE GLOBECOM*, November 1995.

[7] Hyoung-Kee Choi and John O. Limb. A Behavioral Model of Web Traffic. In *International Conference of Networking Protocol ´99 (ICNP 99)*, September 1999. Also available via URL http://users.ece.gatech.edu/~hkchoi.

[8] Edit Cohen, Balachander Krishnamurthy and Jennifer Rexford. Efficient Algorithms for Predicting Requests to Web Servers. Technical Report 981221-01, AT&T Labs – Research, December 1998.

[9] Bruce R. Cunha, Azer Bestavros and Mark E. Crovella. Characteristics of WWW Client-based Traces. Technical Report BU-CS-95-010, Boston University Computer Science Department, July 1995.

[10] N. Desaulniers-Soucy and A. Iuoras. Traffic Modeling with Universal Multifractals. In *Proceedings of GLOBECOM´99*, pp. 1058-1065, 1999.

[11] Hans-Erik Eriksson. Programutveckling med Java. ISBN 91-44-00219-X. Studentlitteratur 1997.

[12] Fredrik Jones. WAP gör Internet mobilt. In PC+, October 1999.

[13] Venkata K. Josyula, Richard B. Bunt and Janelle J. Harms. Measurements of TCP Performance over Wireless Connections. Available via URL http://www.cs.usask.ca/projects/discus/discus_reports.html since 1996.

[14] J. Judge, H.W.P. Beadle and J. Chicharo. Modeling World-Wide Web Request Traffic. In *Proceedings of IS&T SPIE Multimedia Computing and Networking*, San Jose, January 1999.

[15] K. Kant. On Aggregate Traffic Generation with Multifractal Properties. In *Proceedings of GLOBECOM´99*, pp. 1179-1183, 1999.

[16] Reiner Kriesten, Ulrich Kaage, Friedrich Jondral. A Unifying View to Fractional Modeling. In *Proceedings of GLOBECOM'99*, pp. 221-226, 1999.

[17] Andy Liaw and Dick Crawford. Gnuplot 3.5 User's Guide. Available via URL http://www.cs.dartmouth.edu/gnuplot_info.html. November 1994.

[18] Set Lonnert. WAP på djupet. In PC+ pp. 31-33, February 2000.

[19] Bruce A. Mah. An Empirical Model of HTTP Network Traffic. In *Proceedings of INFOCOM ´97*, Kobe, Japan, April 1997.

[20] Motorola, Inc. Available via URL http://www.motorola.com.

[21] Timothy D. Neame, Moshe Zukerman and Ronald G. Addie. Modeling Broadband Traffic Streams. In *Proceedings of GLOBECOM´99*, pp. 1048-1052, 1999.

[22] Nokia. Available via URL http://www.nokia.com.

[23] Steve Park. Lecture Notes On SIMULATION. Department of Computer Science, College of William and Mary. Version 3.1 August 1991.

[24] Vern Paxson and Sally Floyd. Why We Don't Know How To Simulate The Internet. In *Proceedings of the 1997 Winter Simulation Conference*, Atlanta, GA, 1997.

[25] Phone.com, Inc (Unwired Planet). Available via URL http://www.phone.com.

[26] Pradeep Ramakrishnan. Self-Similar Traffic Models. Technical Research Report CSHCN T.R.99-5. Available via URL http://www.isr.umd.edu/CSHCN/ since 1999.

[27] Jan Skansholm. Java direkt. ISBN 91-44-00810-4. Studentlitteratur 1998.

[28] Telefonaktiebolaget LM Ericsson. Available via URL http://www.ericsson.se.

[29] Kerstin Vännman. Matematisk statistik. ISBN 91-44-32181-3. Studentlitteratur 1990.

[30] WAP Forum Ltd. Available via URL http://www.wapforum.org.

[31] WAP Forum. Wireless Application Protocol Architecture Specification. Available via URL http://www.wapforum.org/what/technical.htm since 1998.

[32] WAP Forum. Wireless Application Protocol Wireless Application Environment Overview. Available via URL http://www.wapforum.org/what/technical.htm since 1998.

[33] WAP Forum. W@P™ White Paper. Available via URL http://www.wapforum.org/what/whitepapers.htm since October 1999.

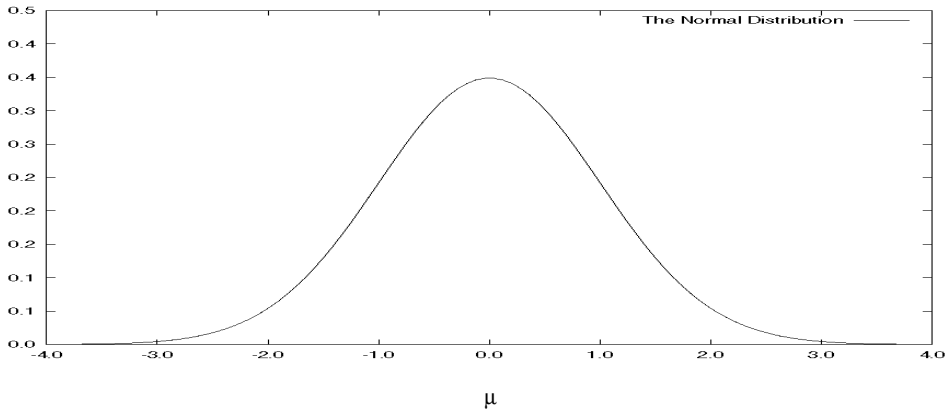
A Abbreviations

CGI	Common Gateway Interface
CPU	Central Processing Unit
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
OSI	Open System Interconnection
PDA	Personal Digital Assistant
RAM	Random Access Memory
ROM	Read Only Memory
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UA-Prof	User Agent Profile
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAP	Wireless Application Protocol
WAE	Wireless Application Environment
WDP	Wireless Datagram Protocol
WGP	Wireless Application Protocol Gateway/Proxy
WML	Wireless Markup Language
WSP	Wireless Session Protocol
WTA	Wireless Telephony Application
WTLS	Wireless Transaction Layer Security
WTP	Wireless Transaction Protocol
XML	eXtensible Markup Language

B Statistical Distributions

Appendix B is intended for those who wish to get familiar with the mathematical formulas of the distributions mentioned in this paper. Only definitions are given, for further reading we recommend [23, 29].

B.1 Definition of the Normal Distribution



$$f(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right) \quad \text{for} -\infty < y < \infty$$

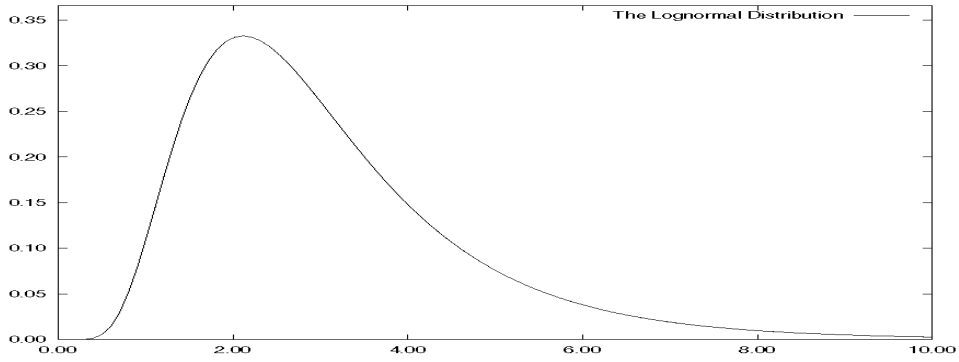
y is the measured parameter.

f**(y)** is the probability of measuring y.

μ is the most probable value of the measured set. f(y) is symmetric around μ.

σ is the standard deviation of the measured set.

B.2 Definition of the Lognormal Distribution



$$f(y) = \frac{1}{y-\theta}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\left(\frac{\log(y-\theta)-\zeta}{\sigma}\right)^2\right) \quad \text{for } y > \theta$$

y is the measured parameter.

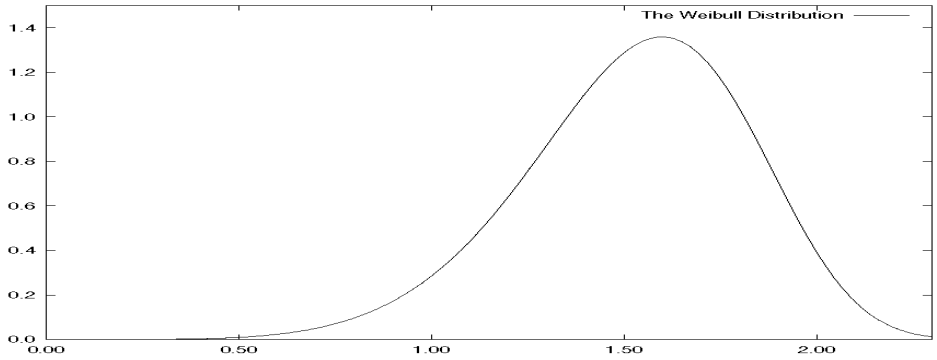
f**(y)** is the probability of measuring y.

θ is the threshold parameter.

ζ is the scale parameter.

σ is the standard deviation of the measured set.

B.3 Definition of the Weibull Distribution



$$f(y) = \frac{c}{\sigma}\left(\frac{y-\theta}{\sigma}\right)^{c-1} \exp\left(-\left(\frac{y-\theta}{\sigma}\right)^{c}\right) \quad \text{for } y > \theta, c > 0$$

y is the measured parameter.

f**(y)** is the probability of measuring y.

θ is the threshold parameter.

ζ is the scale parameter.

c is the shape parameter.

σ is the standard deviation of the measured set.

C Program Code

In Appendix C we will present the four JAVA programs we used to filter out interesting data from the logs. An explanation to the parameters they expect will be given in each subsection. If any questions arise concerning the JAVA code, we refer to [11, 27].

C.1 LogParser

LogParser is a JAVA program with a text-driven interface. As incoming parameters it expects two filenames in the format:

<Existing log file> <New resulting file>

The *existing log file* must have the syntax described in Subsection 4.1.1. The *new resulting file* will have the following appearance when parsed by LogParser: *<Time> <HTTP Requests/Sessions>*.

```
//Program code for LogParser:

import java.io.*;
import java.util.*;
import java.awt.*;
import java.applet.*;
import java.lang.*;

public class LogParser
{
  public void parse(String inFile, String outFile) throws
  IOException
  {
          StringTokenizer st, newTime;
          String inputLine, day, month, date, time, year, number,
          yTime1, yTime2, yTime;
          int totalNumber = 0, totalEntries = 0 ;

          try
          {
              BufferedReader inBuffer = new BufferedReader(new
              FileReader(new File(inFile)));
              BufferedWriter outBuffer = new BufferedWriter(new
              FileWriter(new File(outFile)));

              while((inputLine = inBuffer.readLine()) != null)
              {
                inputLine.trim();
                st = new StringTokenizer(inputLine, " ");
                day = st.nextToken();
```

```
                month = st.nextToken();
                date = st.nextToken();
                time = st.nextToken();
                year = st.nextToken();
                number = st.nextToken();

                newTime = new StringTokenizer(time, ":");
                yTime1 = newTime.nextToken();
                Integer tempTime = new Integer(yTime1);
                yTime1 = tempTime.toString();
                yTime2 = newTime.nextToken();
                yTime = yTime1 + "." + yTime2;

                Integer temp = new Integer(number);
                totalNumber += temp.intValue();

                String outputLine = yTime + " " + number;

                outBuffer.write(outputLine); outBuffer.newLine();
                outBuffer.flush();

                totalEntries = totalEntries + 1;
            }
            System.out.println("Total number: " + totalNumber);
            System.out.println("Total entries: " + totalEntries);
        }
        catch(FileNotFoundException e)
        {
                System.err.println("An error occured.");
        }
  }

  static public void main(String[] arg ) throws IOException
  {
        LogParser parser = new LogParser();

        if(arg.length == 2)
                parser.parse(arg[0], arg[1]);
        else
                System.err.println("Too few or too many
                arguments.");
  }
}
```

C.2 RequestParser

RequestParser is a JAVA program with a text-driven interface. As incoming parameters it expects three filenames in the format:

<Existing log file with HTTP Requests> <Existing log file with HTTP Requests> <New resulting file>

The *existing log files with HTTP requests* must have been parsed by LogParser. The *new resulting file* will have the following appearance when parsed by RequestParser, since it will

add the HTTP requests from the *existing log files with HTTP requests*: *<Time> <HTTP Requests>*.

```
//Program code for RequestParser:


import java.io.*;
import java.util.*;
import java.awt.*;
import java.applet.*;
import java.lang.*;

public class RequestParser
{
  public void parse(String inFile1, String inFile2, String outFile)
  throws IOException
  {
          StringTokenizer st1, st2;
          String inputLine1, inputLine2, day, month, date, time,
          year, number1, number2;
          int totalNumber;

          try
          {
              BufferedReader inBuffer1 = new BufferedReader(new
              FileReader(new File(inFile1)));
              BufferedReader inBuffer2 = new BufferedReader(new
              FileReader(new File(inFile2)));
              BufferedWriter outBuffer = new BufferedWriter(new
              FileWriter(new File(outFile)));

              while((inputLine1 = inBuffer1.readLine()) != null &&
              (inputLine2 = inBuffer2.readLine()) != null)
              {
                st1 = new StringTokenizer(inputLine1, " ");
                day = st1.nextToken();
                month = st1.nextToken();
                date = st1.nextToken();
                time = st1.nextToken();
                year = st1.nextToken();
                number1 = st1.nextToken();

                st2 = new StringTokenizer(inputLine2, " ");
                day = st2.nextToken();
                month = st2.nextToken();
                date = st2.nextToken();
                time = st2.nextToken();
                year = st2.nextToken();
                number2 = st2.nextToken();

                Integer temp1 = new Integer(number1);
                Integer temp2 = new Integer(number2);
                totalNumber = temp1.intValue() + temp2.intValue();
```

```
                String outputLine = day+" "+month+" "+date+"
                "+time+" "+year+" "+totalNumber;
                outBuffer.write(outputLine); outBuffer.newLine();
                outBuffer.flush();
             }
          }
          catch(FileNotFoundException e)
          {
                   System.err.println("An error occured.");
          }
  }

  static public  void main(String[] arg ) throws IOException
  {
          RequestParser parser = new RequestParser();

          if(arg.length == 3)
                   parser.parse(arg[0], arg[1], arg[2]);
          else
                   System.err.println("Too few or too many
                   arguments.");
  }
}
```

C.3 LinkParser

LinkParser is a JAVA program with a text-driven interface. As incoming parameters it expects three filenames in the format:

<Existing log file with HTTP Requests> <Existing log file with Sessions> <New resulting file>

LogParser must have parsed *existing log file with sessions* and RequestParser must have parsed *existing log file with HTTP requests*. The *new resulting file* will have the following appearance when parsed by LinkParser: *<Time> <Sessions> <HTTP Requests>*.

```
//Program code for LinkParser:

import java.io.*;
import java.util.*;
import java.awt.*;
import java.applet.*;
import java.lang.*;

public class LinkParser
{
  public void parse(String inFile1, String inFile2, String outFile)
  throws IOException
  {
          StringTokenizer st, st2;
          String inputLine, inputLine2, time, time2, number,
          number2;
```

```
        int totalNumber = 0, totalEntries = 0;

        try
        {
                BufferedReader inBuffer1 = new
                BufferedReader(new FileReader(new
                File(inFile1)));
                BufferedReader inBuffer2 = new
                BufferedReader(new FileReader(new
                File(inFile2)));
                BufferedWriter outBuffer = new
                BufferedWriter(new FileWriter(new
                File(outFile)));

                while(((inputLine = inBuffer1.readLine()) !=
                null) && ((inputLine2 = inBuffer2.readLine()) !=
                null))
                {
                  st = new StringTokenizer(inputLine," ");
                  st2 = new StringTokenizer(inputLine2," ");
                  time = st.nextToken();
                  number = st.nextToken();

                  time2 = st2.nextToken();
                  number2 = st2.nextToken();

                  String outputLine = number2 + " " + time + " "
                  + number;
                  outBuffer.write(outputLine);
                  outBuffer.newLine();
                  outBuffer.flush();
                }
        }
        catch(FileNotFoundException e)
        {
                System.err.println("An error occured.");
        }
  }


  static public void main(String[] arg ) throws IOException
  {
        LinkParser parser = new LinkParser();

        if(arg.length == 3)
                parser.parse(arg[0], arg[1], arg[2]);
        else
                System.err.println("Too few or too many
                arguments.");
  }
}
```

C.4 AverageParser

AverageParser is a JAVA program with a text-driven interface. As incoming parameters it expects three filenames in the format:

<Existing log file with HTTP Requests> <Existing log file with Sessions> <New resulting file>

LogParser must have parsed *existing log file with sessions* and RequestParser must have parsed *existing log file with HTTP requests*. The *new resulting file* will have the following appearance when parsed by AverageParser: *<Time> <HTTP Requests per Session>*.

```
//Program code for AverageParser:

import java.io.*;
import java.util.*;
import java.awt.*;
import java.applet.*;
import java.lang.*;

public class AverageParser
{

  public void parse(String inFile1, String inFile2, String outFile)
throws IOException
  {
          StringTokenizer st1, st2;
          String inputLine1, inputLine2, time, number1, number2;
          float average, totalAverage = 0;
          int entries = 0;

          try
          {
              BufferedReader inBuffer1 = new BufferedReader(new
              FileReader(new File(inFile1)));
              BufferedReader inBuffer2 = new BufferedReader(new
              FileReader(new File(inFile2)));
              BufferedWriter outBuffer = new BufferedWriter(new
              FileWriter(new File(outFile)));

              while((inputLine1 = inBuffer1.readLine()) != null &&
              (inputLine2 = inBuffer2.readLine()) != null)
              {
                st1 = new StringTokenizer(inputLine1, " ");
                time = st1.nextToken();
                number1 = st1.nextToken();

                st2 = new StringTokenizer(inputLine2, " ");
                time = st2.nextToken();
                number2 = st2.nextToken();

                Float temp1 = new Float(number1);
                Float temp2 = new Float(number2);

                if(temp2.floatValue() == 0)
```

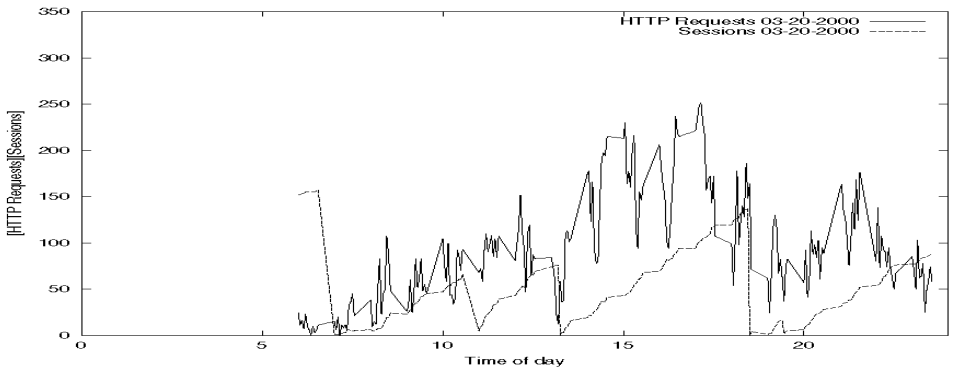
```
                {
                    average = 0;
                }
                else
                    average = temp1.floatValue() /
                    temp2.floatValue();

              totalAverage += average;
              entries++;

              String outputLine = time+" "+average;

              outBuffer.write(outputLine); outBuffer.newLine();
              outBuffer.flush();
            }
            if(entries != 0)
            {
                System.out.println("Total avg: " +totalAverage);
                System.out.println("Entries: " +entries);
                totalAverage = totalAverage / entries;
                System.out.println("Sum: " +totalAverage);
            }
         }
         catch(FileNotFoundException e)
         {
            System.err.println("An error occured.");
         }
   }

   static public  void main(String[] arg ) throws IOException
   {
           AverageParser parser = new AverageParser();

           if(arg.length == 3)
                    parser.parse(arg[0], arg[1], arg[2]);
           else
                    System.err.println("Too many or too few
                    arguments.");
   }
}
```

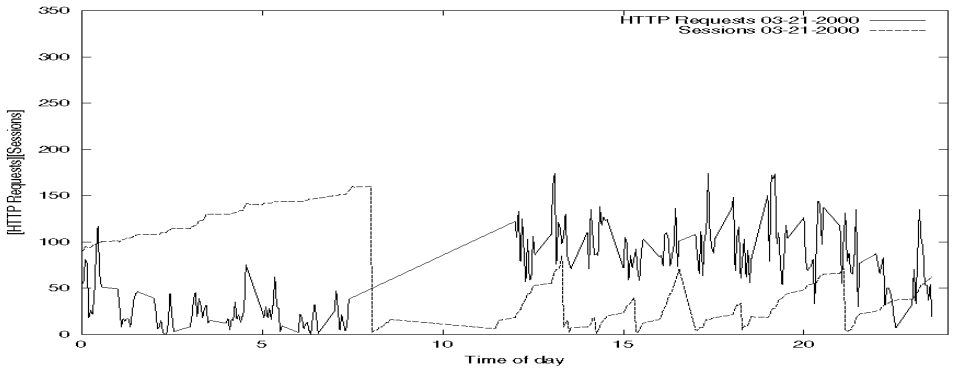
D Graphs

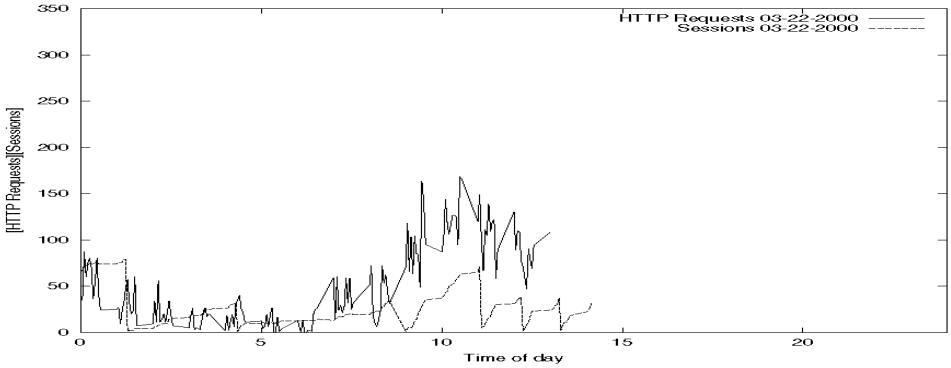
The content of Appendix D is the total amount of graphs produced in this project. Each subsection takes up different aspects of the measured dataset.

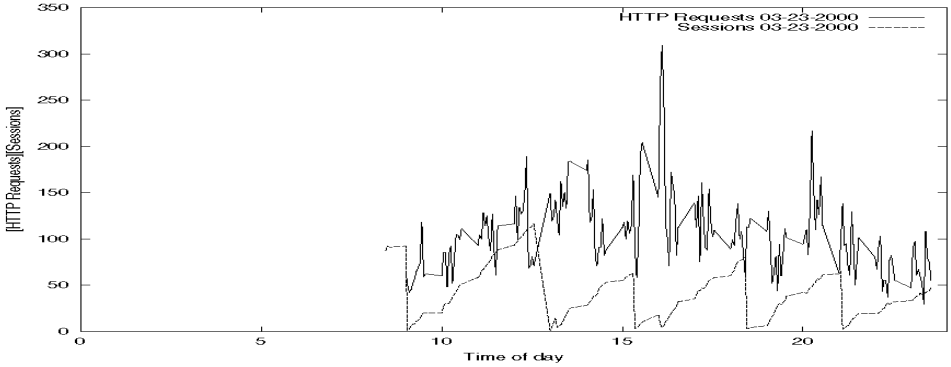
D.1 Observed Sessions and HTTP Requests

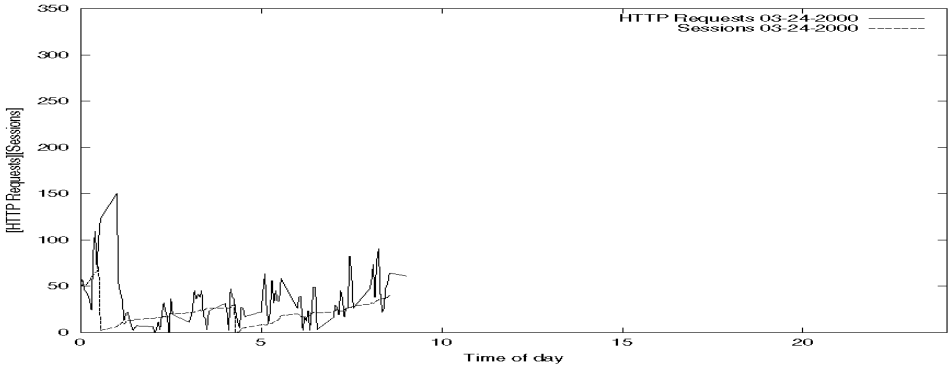
Here we present the twelve measured days in graphs showing the number of sessions and the number of HTTP requests for each day. The label on each graph will contain enough information to make the graphs self-explanatory. Some of the graphs may appear odd, which partly can be explained by insufficient data.

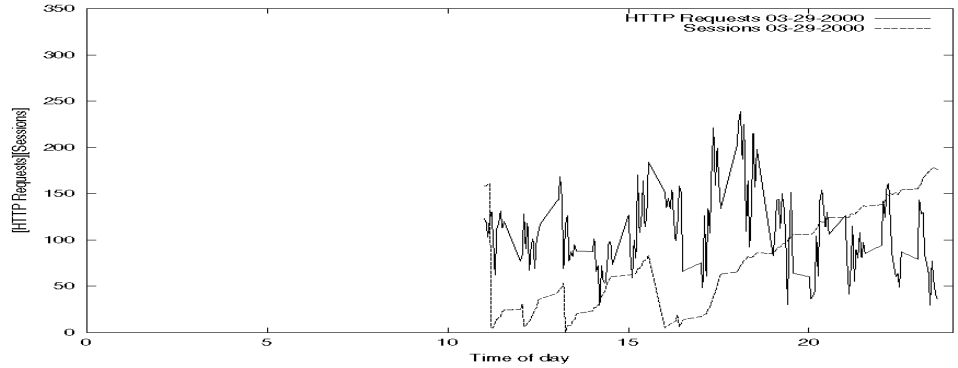


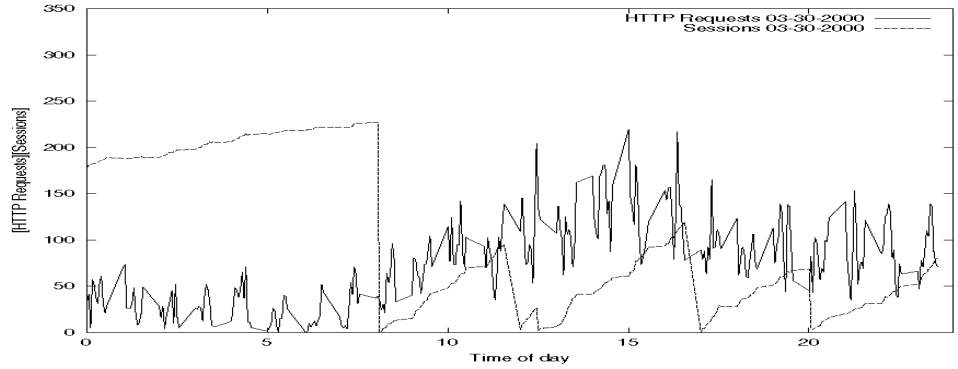


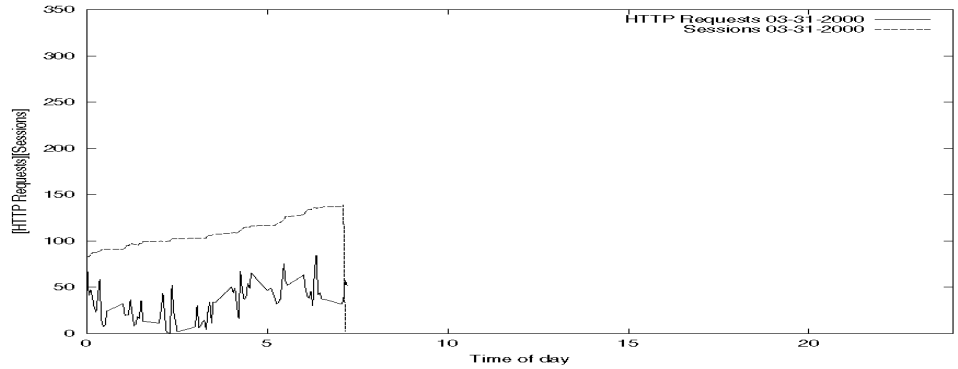

HTTP Requests 03-22-2000
Sessions 03-22-2000
[HTTP Requests][Sessions]
Time of day
0
50
100
150
200
250
300
350
0
5
10
15
20

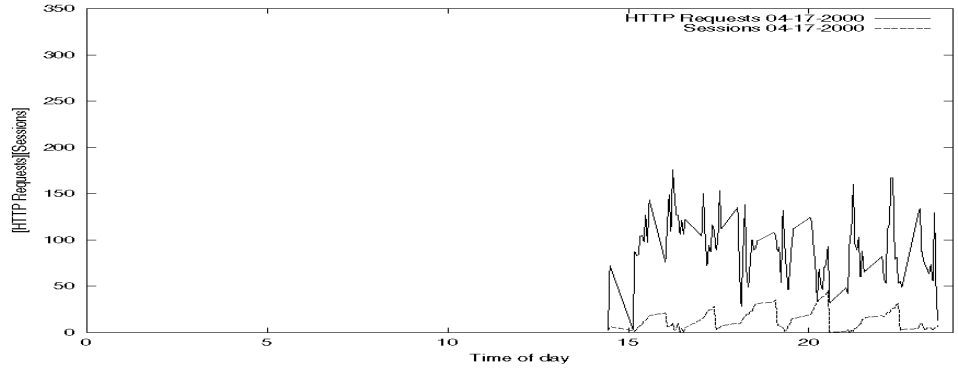

HTTP Requests 03-23-2000
Sessions 03-23-2000
[HTTP Requests][Sessions]
Time of day
0
50
100
150
200
250
300
350
0
5
10
15
20

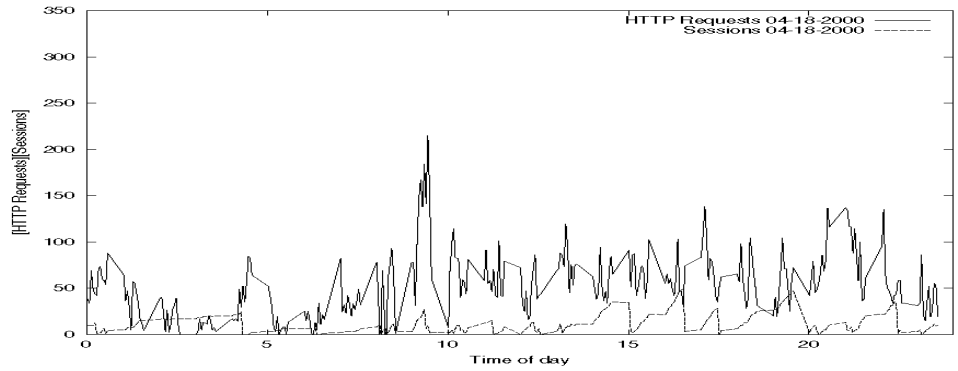

HTTP Requests 03-24-2000
Sessions 03-24-2000
[HTTP Requests][Sessions]
Time of day
0
50
100
150
200
250
300
350
0
5
10
15
20

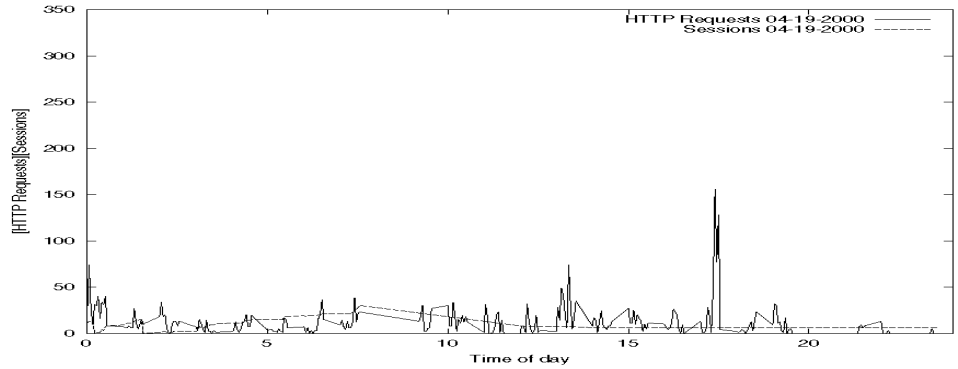

HTTP Requests 03-29-2000
Sessions 03-29-2000
[HTTP Requests][Sessions]
Time of day
0
50
100
150
200
250
300
350
0
5
10
15
20

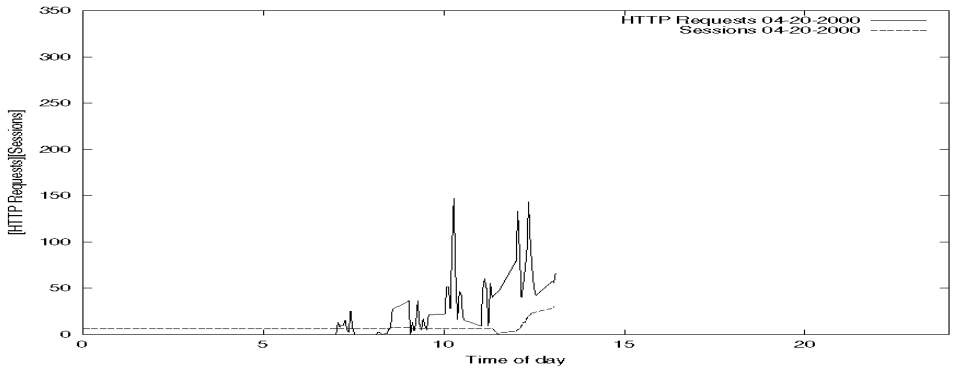

HTTP Requests 03-30-2000
Sessions 03-30-2000
[HTTP Requests][Sessions]
Time of day
0
50
100
150
200
250
300
350
0
5
10
15
20


HTTP Requests 03-31-2000
Sessions 03-31-2000
[HTTP Requests][Sessions]
Time of day
0
50
100
150
200
250
300
350
0
5
10
15
20


HTTP Requests 04-17-2000
Sessions 04-17-2000
[HTTP Requests][Sessions]
Time of day
0
50
100
150
200
250
300
350
0
5
10
15
20

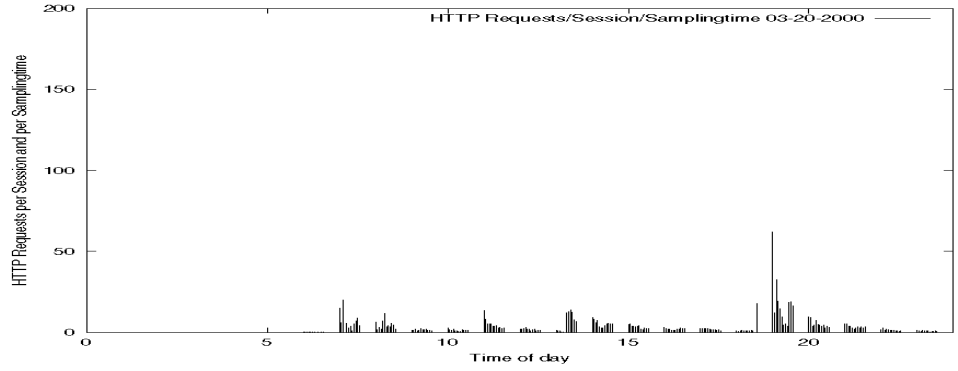

HTTP Requests 04-18-2000
Sessions 04-18-2000
[HTTP Requests][Sessions]
Time of day
0
50
100
150
200
250
300
350
0
5
10
15
20

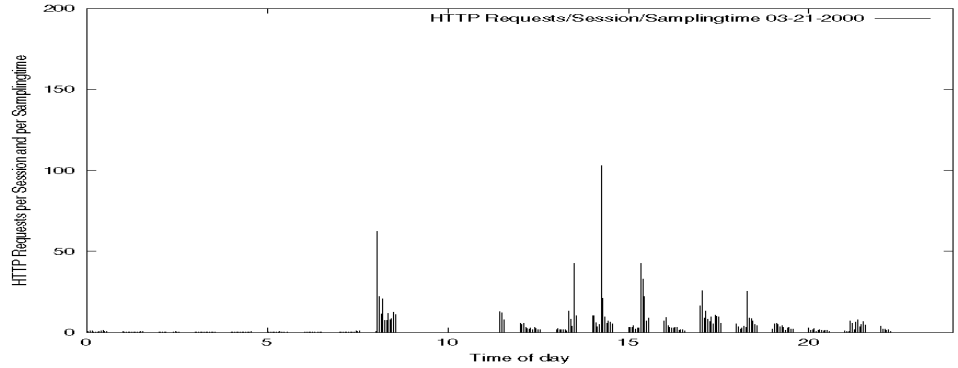

HTTP Requests 04-19-2000
Sessions 04-19-2000
[HTTP Requests][Sessions]
Time of day
0
50
100
150
200
250
300
350
0
5
10
15
20

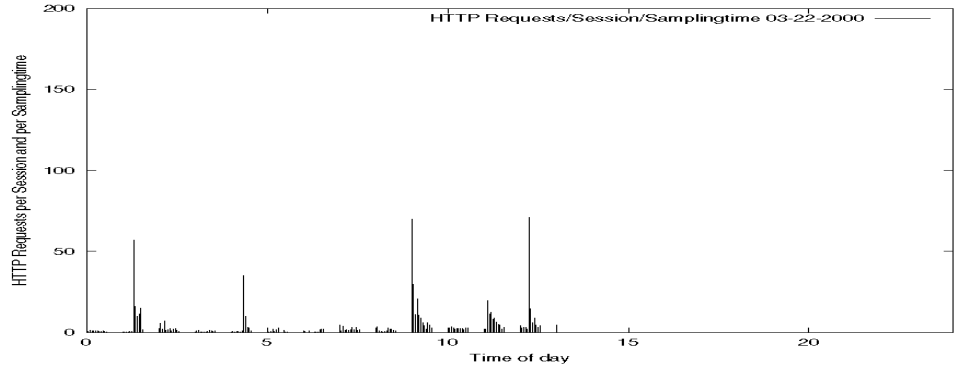


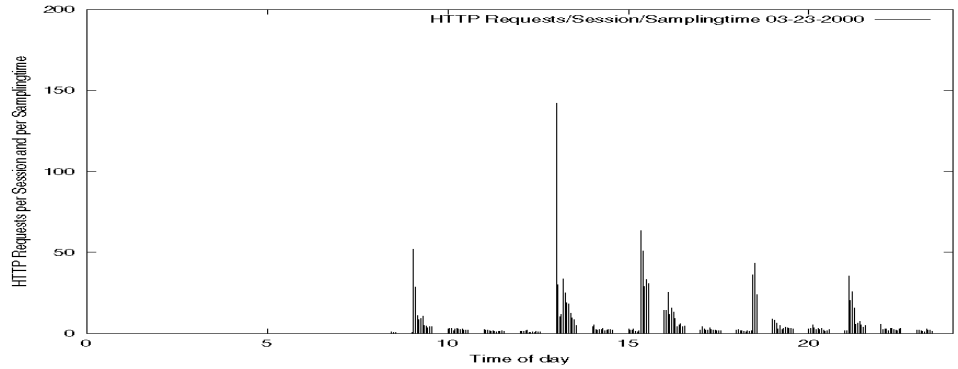
D.2 The Request Rate per Sampling Time

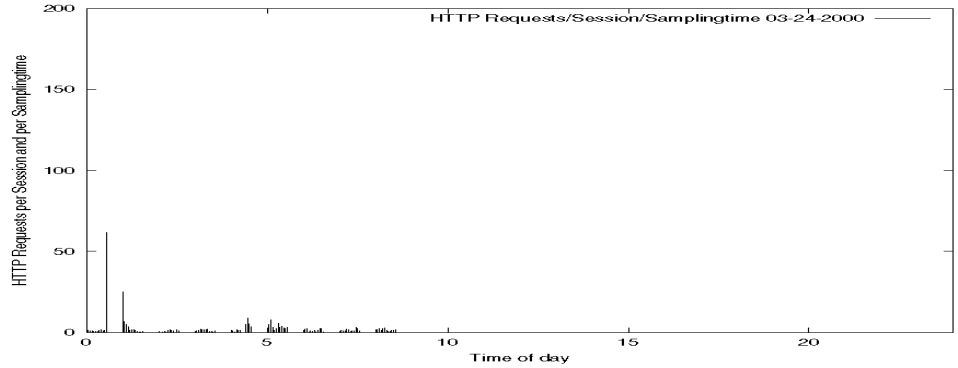
Here we present the twelve measured days in graphs showing the request rate per sampling time for each day, that is the number of HTTP requests per session per sampling time throughout a day. The two last graphs will show the request rates for eight specifically chosen days and the request rates for all twelve days. The label on each graph will contain enough information to make the graphs self-explanatory. Some of the graphs may appear odd, which partly can be explained by insufficient data. The regular empty spaces in each of these graphs are, as far as we have been able to understand, due to Gnuplot's [17] implementation of impulses.

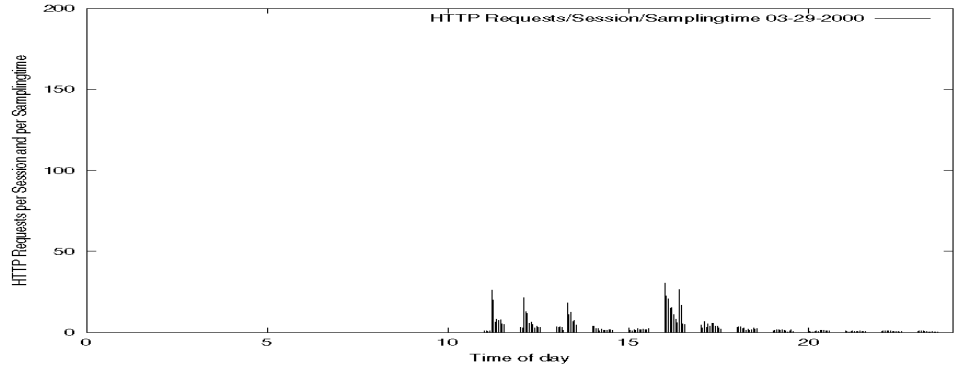


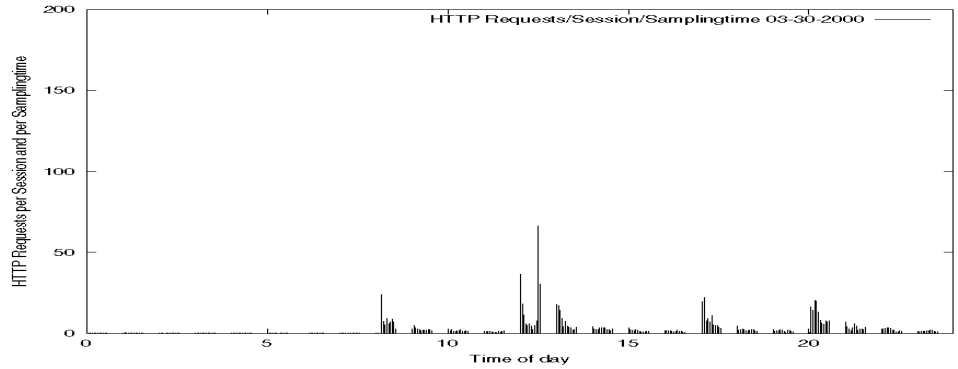

HTTP Requests/Session/Samplingtime 03-21-2000
HTTP Requests per Session and per Samplingtime
Time of day
0
50
100
150
200
0
5
10
15
20

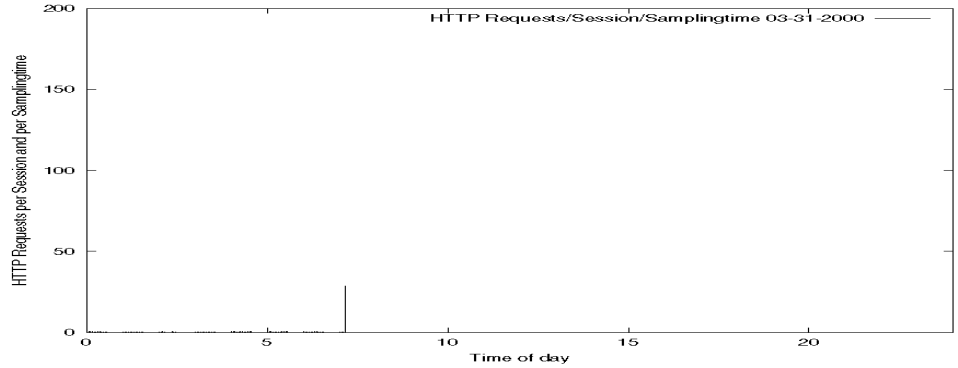

HTTP Requests/Session/Samplingtime 03-22-2000
HTTP Requests per Session and per Samplingtime
Time of day
0
50
100
150
200
0
5
10
15
20

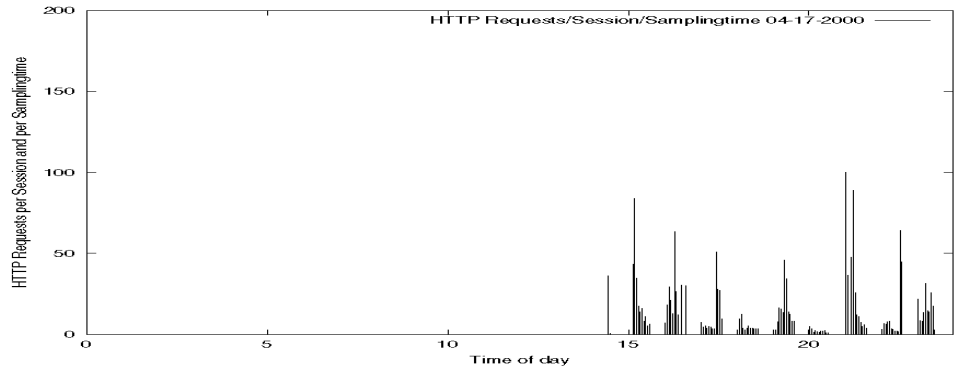

HTTP Requests/Session/Samplingtime 03-23-2000
HTTP Requests per Session and per Samplingtime
Time of day
0
50
100
150
200
0
5
10
15
20

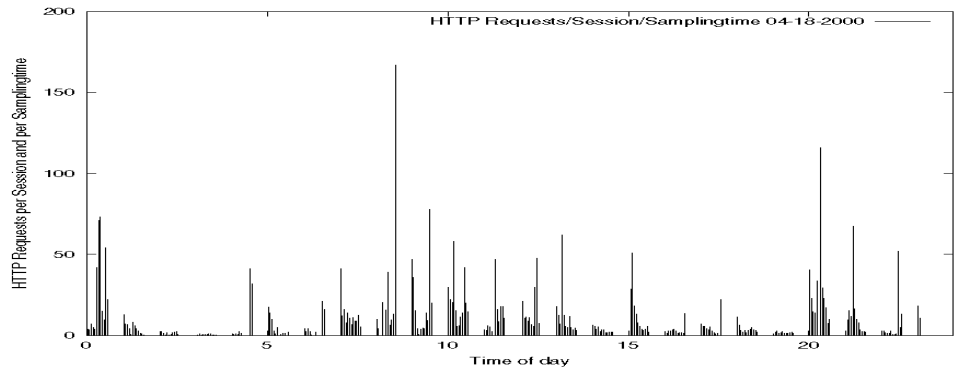

HTTP Requests/Session/Samplingtime 03-24-2000
HTTP Requests per Session and per Samplingtime
Time of day
0
50
100
150
200
0
5
10
15
20

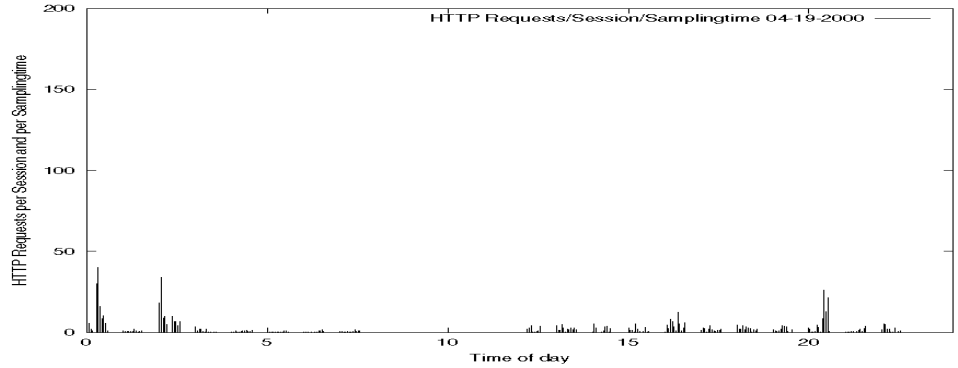

HTTP Requests/Session/Samplingtime 03-29-2000
HTTP Requests per Session and per Samplingtime
Time of day
0
50
100
150
200
0
5
10
15
20

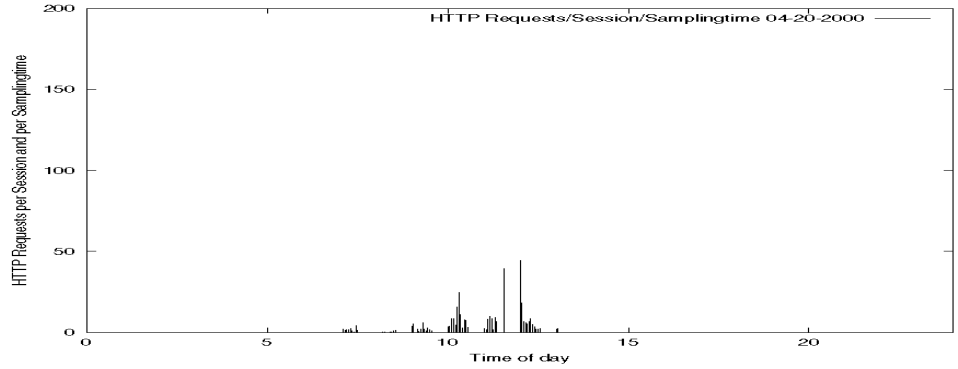

HTTP Requests/Session/Samplingtime 03-30-2000
HTTP Requests per Session and per Samplingtime
Time of day
0
50
100
150
200
0
5
10
15
20

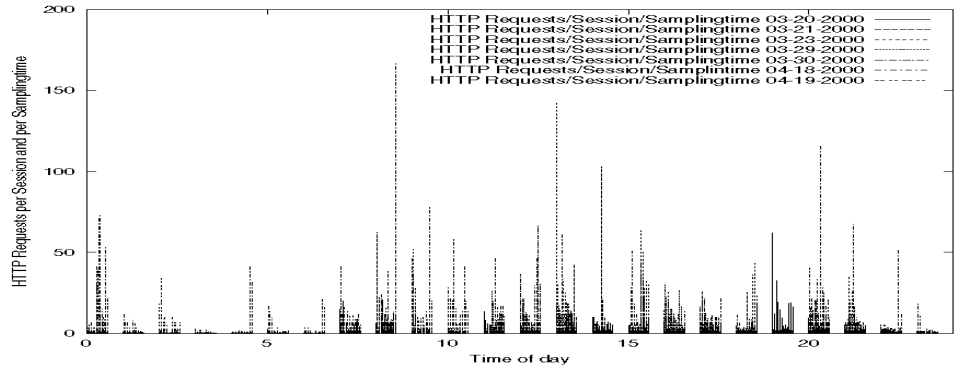

HTTP Requests/Session/Samplingtime 03-31-2000
HTTP Requests per Session and per Samplingtime
Time of day
0
50
100
150
200
0
5
10
15
20

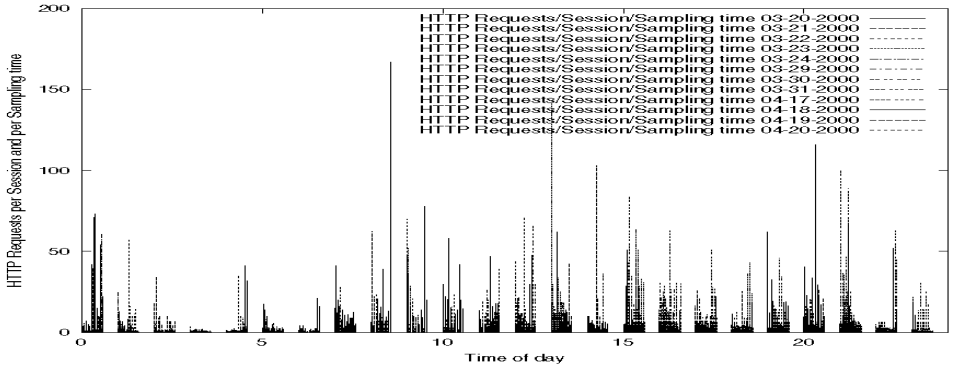

HTTP Requests/Session/Samplingtime 04-17-2000
HTTP Requests per Session and per Samplingtime
Time of day
0
50
100
150
200
0
5
10
15
20


HTTP Requests/Session/Samplingtime 04-18-2000
HTTP Requests per Session and per Samplingtime
Time of day
0
50
100
150
200
0
5
10
15
20


HTTP Requests/Session/Samplingtime 04-19-2000
HTTP Requests per Session and per Samplingtime
Time of day
0
50
100
150
200
0
5
10
15
20

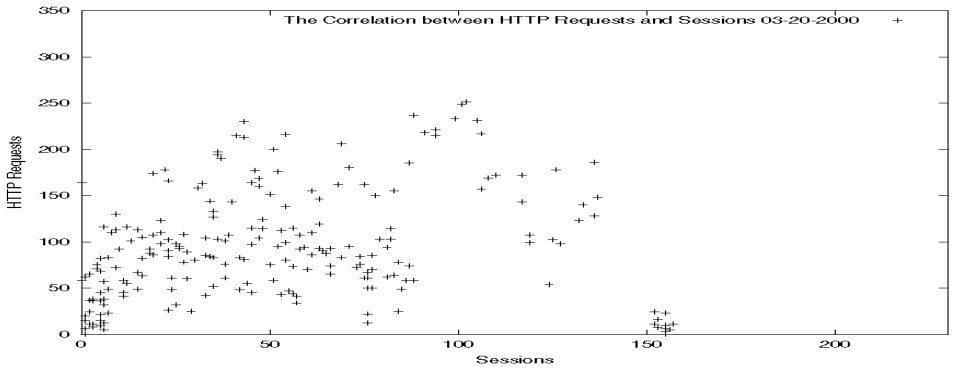

HTTP Requests/Session/Samplingtime 04-20-2000
HTTP Requests per Session and per Samplingtime
Time of day
0
50
100
150
200
0
5
10
15
20

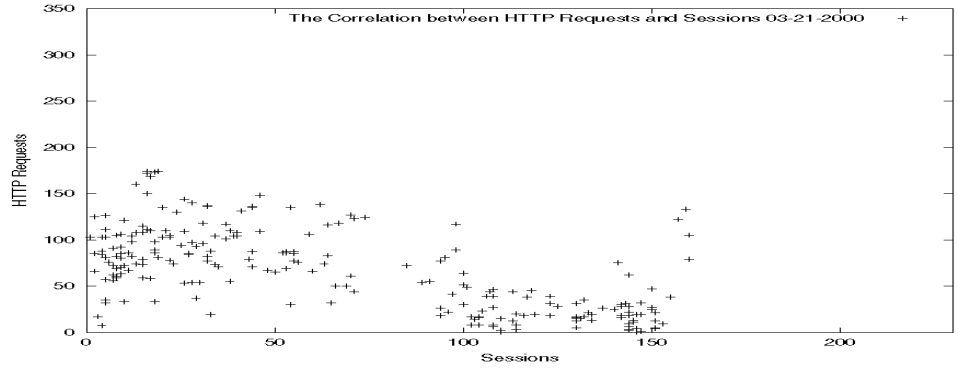

HTTP Requests/Session/Samplingtime 03-20-2000
HTTP Requests/Session/Samplingtime 03-21-2000
HTTP Requests/Session/Samplingtime 03-23-2000
HTTP Requests/Session/Samplingtime 03-29-2000
HTTP Requests/Session/Samplingtime 03-30-2000
HTTP Requests/Session/Samplingtime 04-18-2000
HTTP Requests/Session/Samplingtime 04-19-2000
HTTP Requests per Session and per Samplingtime
Time of day
0
50
100
150
200
0
5
10
15
20

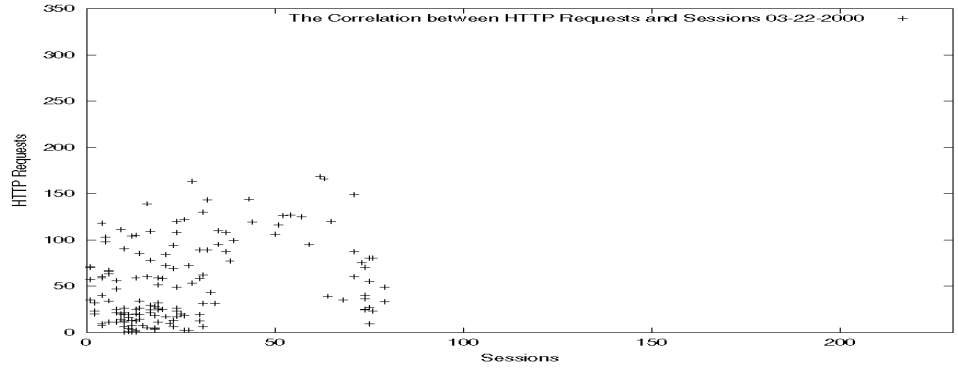


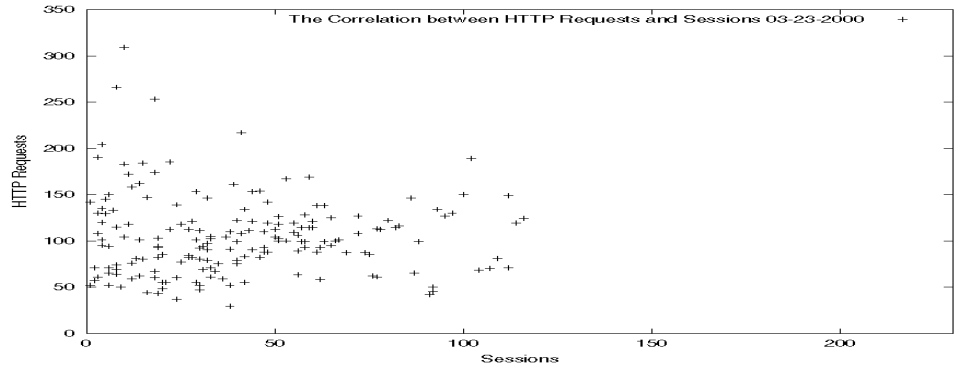
D.3 The Correlation between Sessions and HTTP Requests

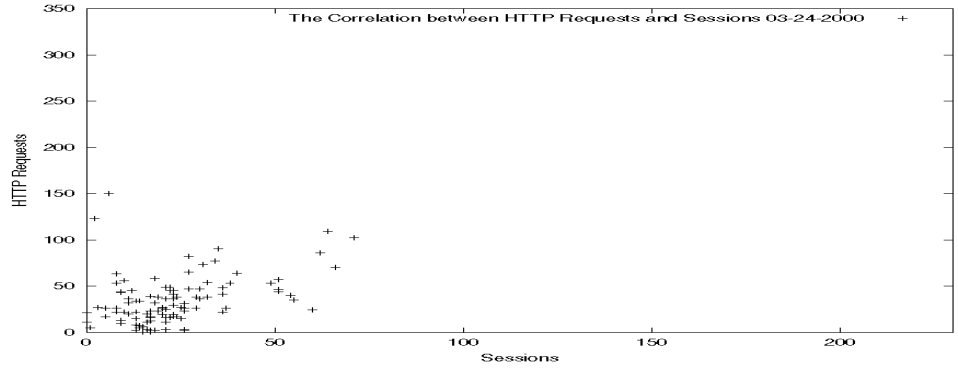
Here we present the twelve measured days in graphs showing the possible correlation between sessions and HTTP requests for each day. In this subsection the graphs attempt to display the given correlation without the time parameter. The last graph shows the correlation for seven specifically chosen days. The label on each graph will contain enough information to make the graphs self-explanatory. Some of the graphs may appear odd, which partly can be explained by insufficient data.

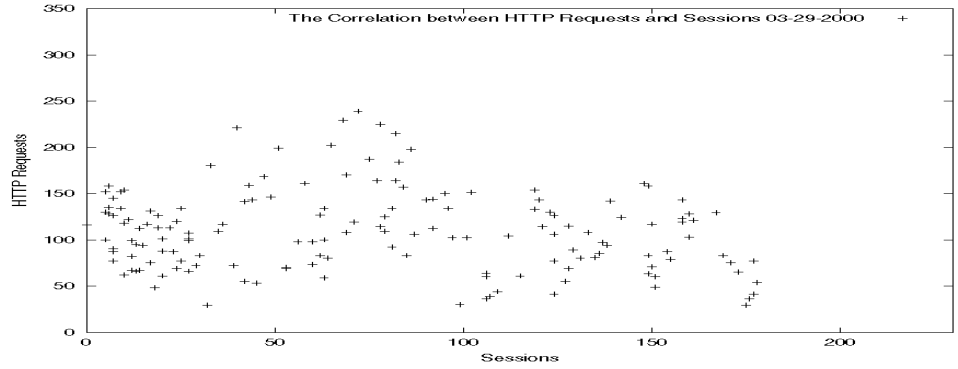


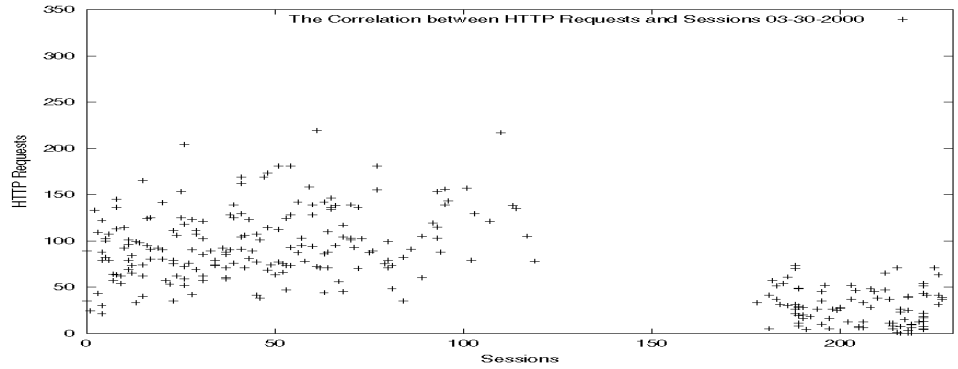

The Correlation between HTTP Requests and Sessions 03-21-2000
HTTP Requests
Sessions
0
50
100
150
200
250
300
350

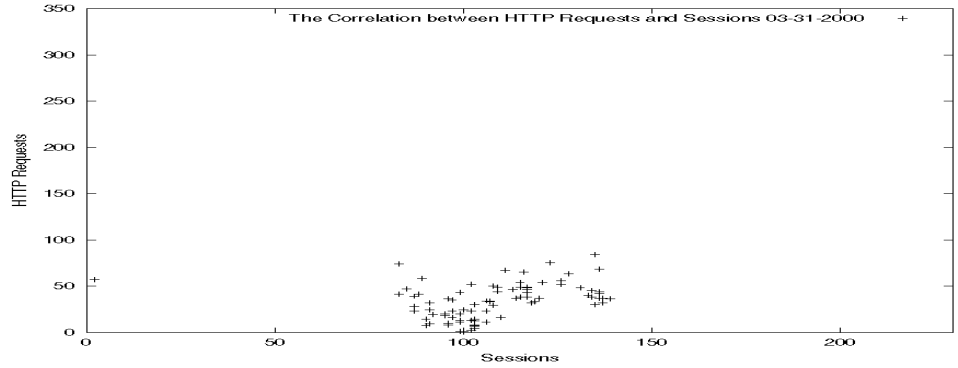

The Correlation between HTTP Requests and Sessions 03-22-2000
HTTP Requests
Sessions
0
50
100
150
200
250
300
350

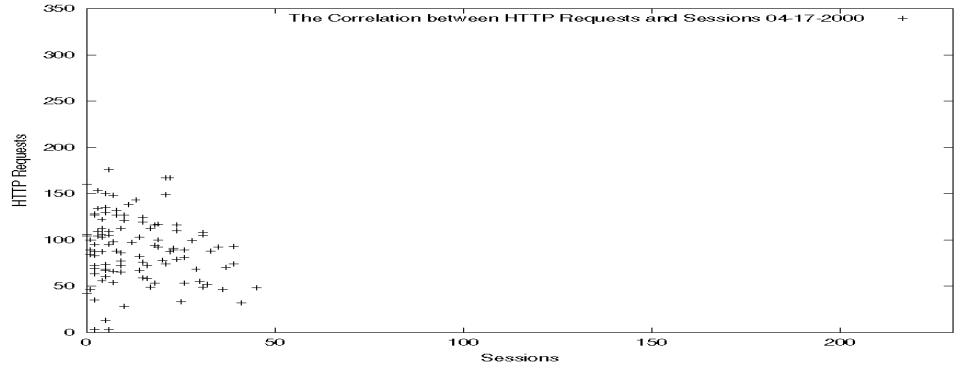

The Correlation between HTTP Requests and Sessions 03-23-2000
HTTP Requests
Sessions
0
50
100
150
200
250
300
350

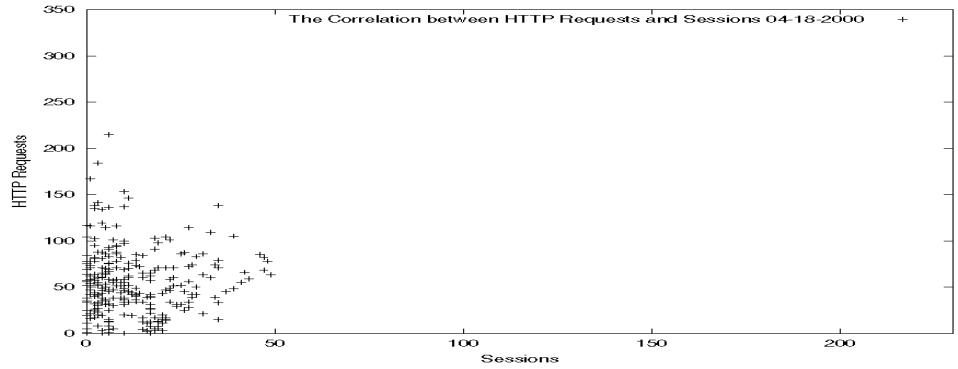

The Correlation between HTTP Requests and Sessions 03-24-2000
HTTP Requests
Sessions
0
50
100
150
200
250
300
350

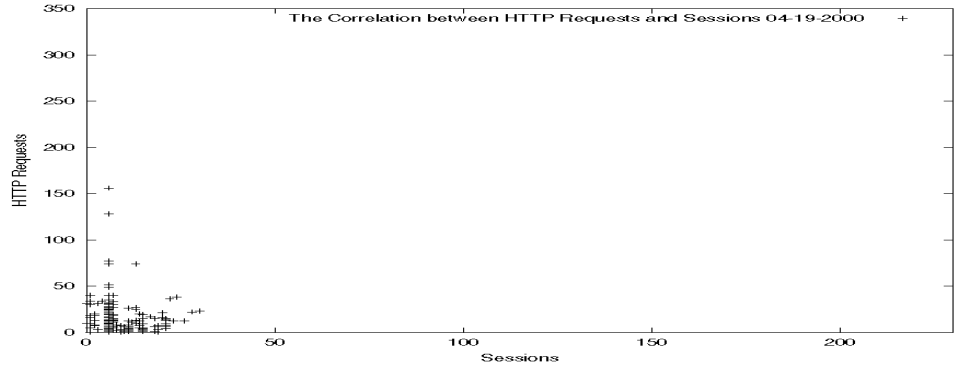

The Correlation between HTTP Requests and Sessions 03-29-2000
HTTP Requests
Sessions
0
50
100
150
200
250
300
350

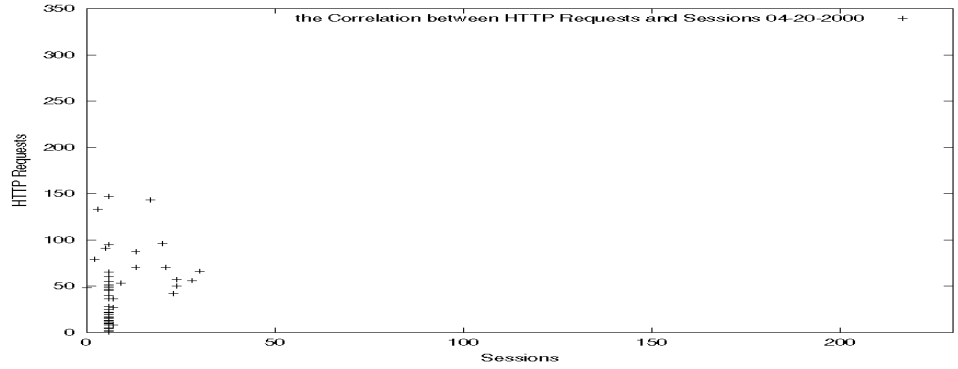

The Correlation between HTTP Requests and Sessions 03-30-2000
HTTP Requests
Sessions
0
50
100
150
200
250
300
350

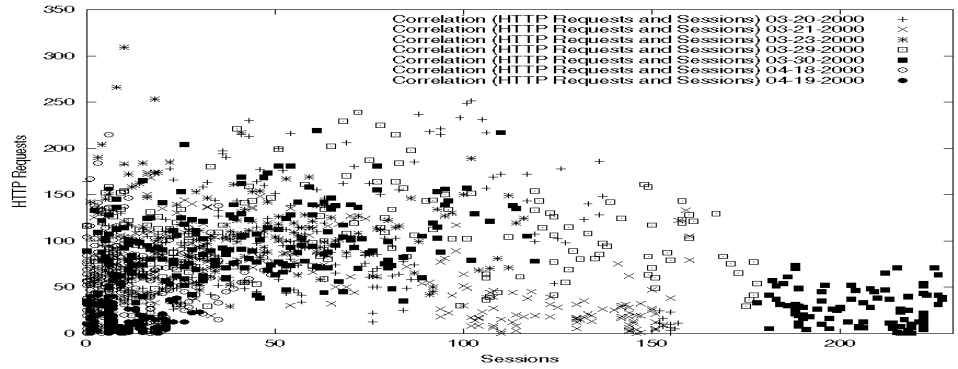

The Correlation between HTTP Requests and Sessions 03-31-2000
HTTP Requests
Sessions
0
50
100
150
200
250
300
350


The Correlation between HTTP Requests and Sessions 04-17-2000
HTTP Requests
Sessions
0
50
100
150
200
250
300
350


The Correlation between HTTP Requests and Sessions 04-18-2000
HTTP Requests
Sessions
0
50
100
150
200
250
300
350



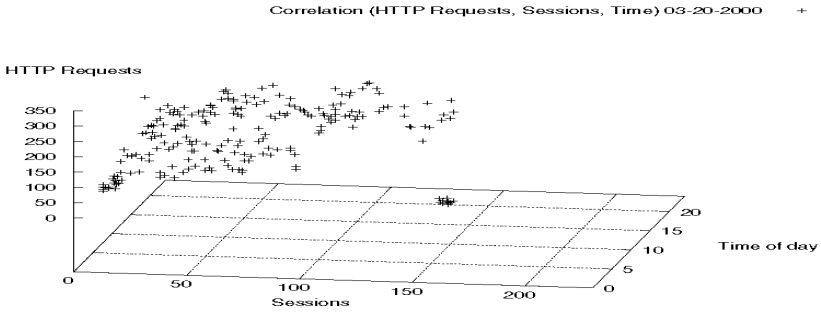


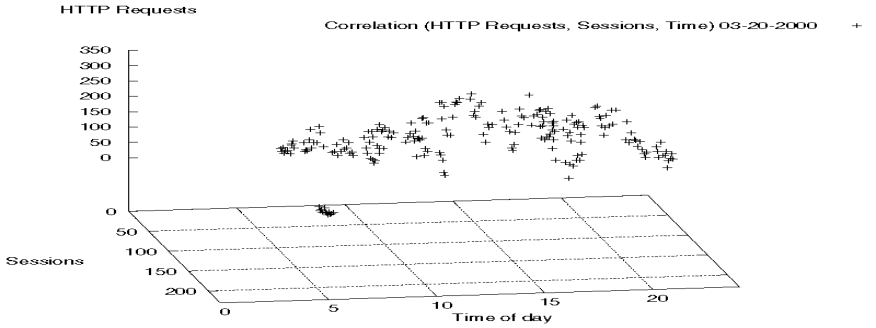


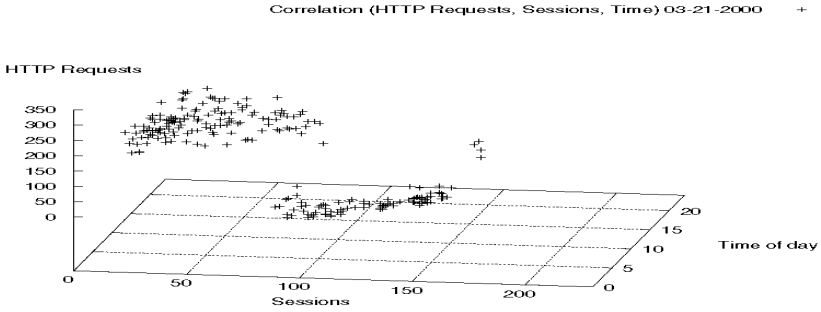
D.4 The Correlation between Sessions and HTTP Requests over Time

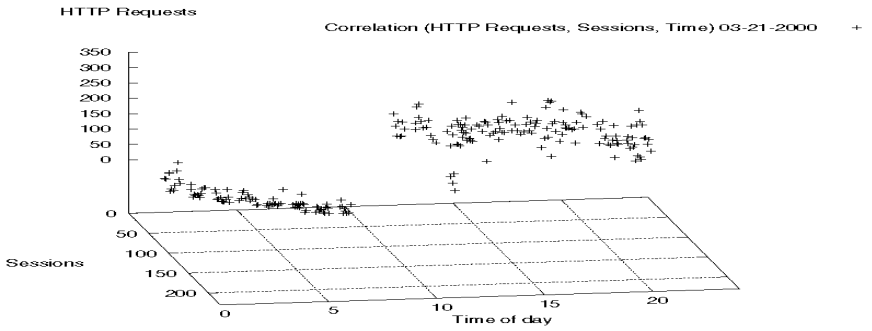
Here we present the twelve measured days in graphs showing the possible correlation between sessions and HTTP requests, with concern to time, for each day, that is in three-dimensional graphs. Every correlation will be shown from two angles except the last ones that will be shown from three. The last graphs show the correlation for seven specifically chosen days in the form of points and impulses. The label on each graph will contain enough

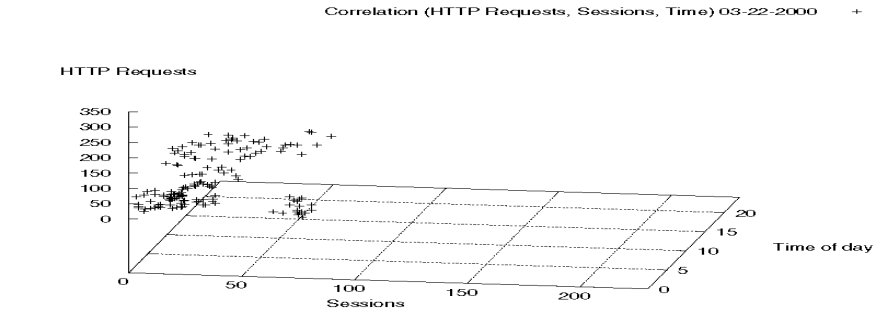
information to make the graphs self-explanatory. Some of the graphs may appear odd, which partly can be explained by insufficient data.

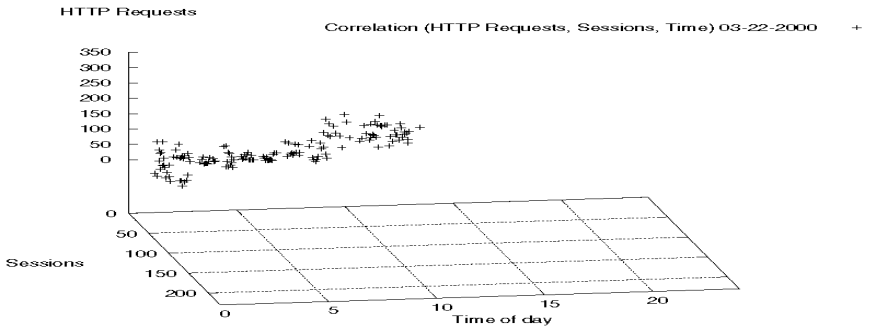


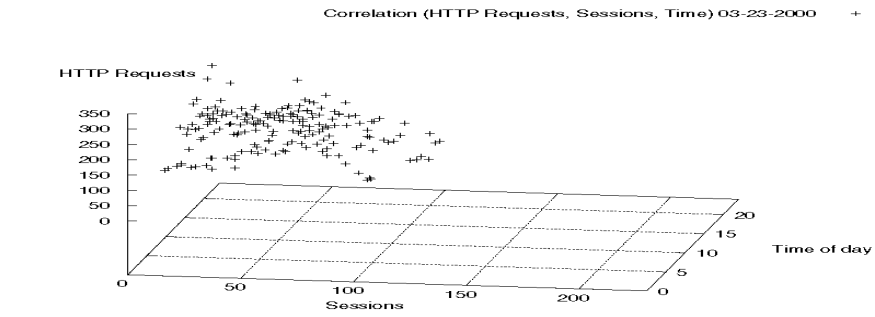


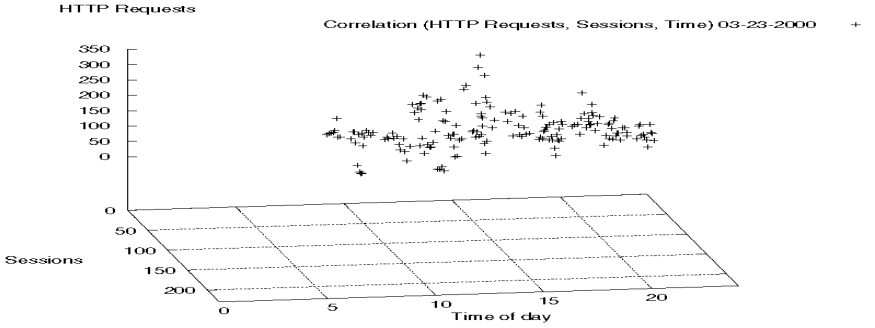


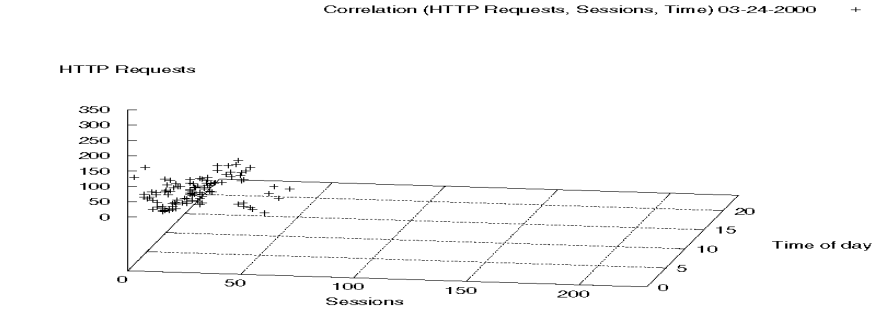

HTTP Requests
Correlation (HTTP Requests, Sessions, Time) 03-21-2000
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

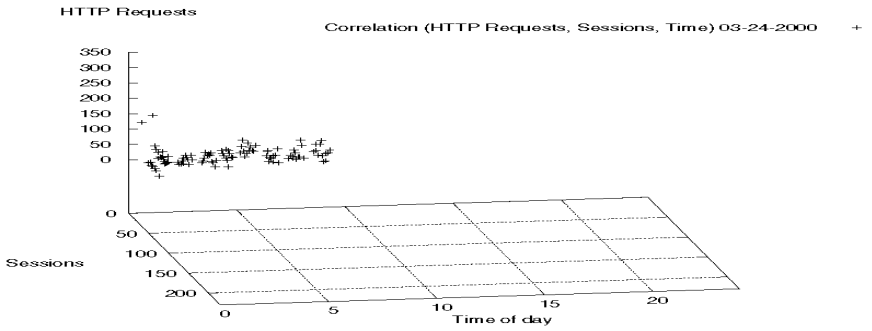

Correlation (HTTP Requests, Sessions, Time) 03-22-2000
HTTP Requests
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
20
15
10
5
0
Time of day

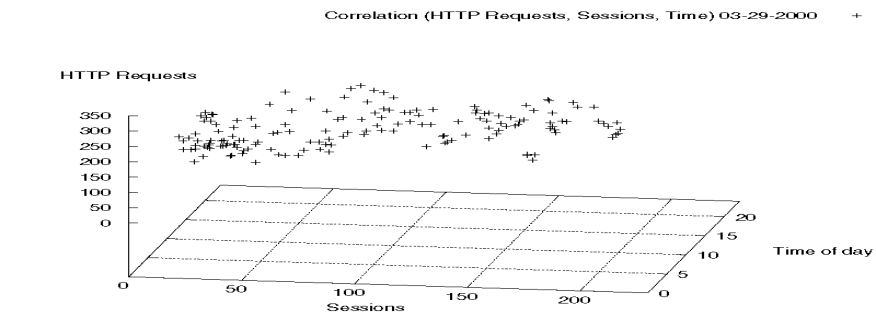

HTTP Requests
Correlation (HTTP Requests, Sessions, Time) 03-22-2000
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

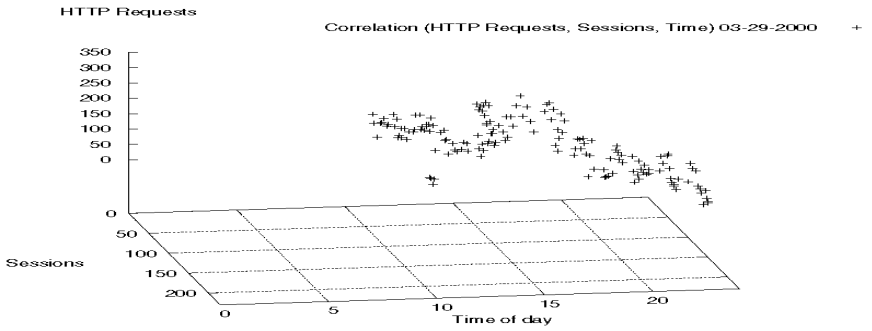

Correlation (HTTP Requests, Sessions, Time) 03-23-2000
HTTP Requests
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

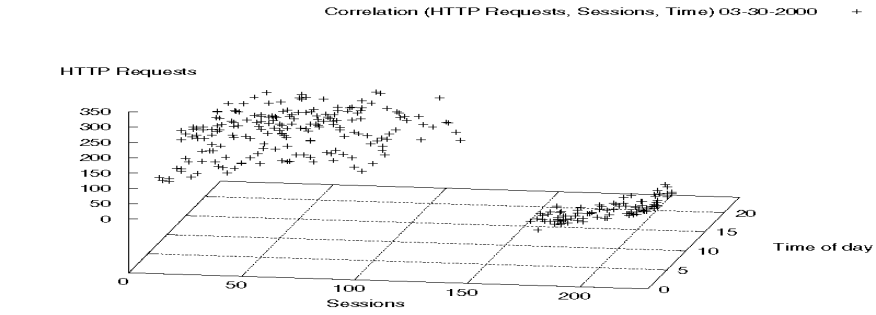

HTTP Requests
Correlation (HTTP Requests, Sessions, Time) 03-23-2000
350
300
250
200
150
100
50
0
Sessions
0
50
100
150
200
0
5
10
15
20
Time of day

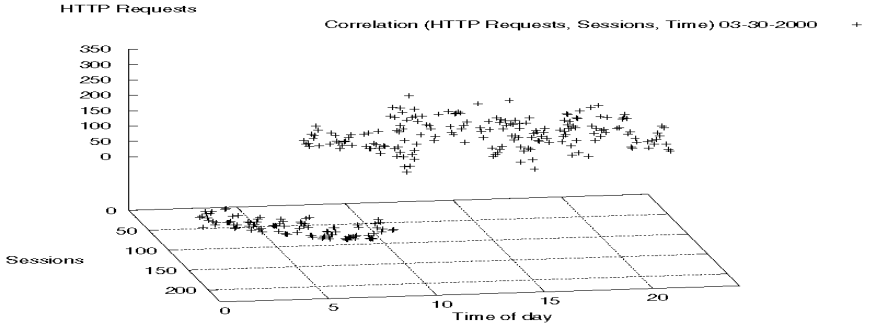

Correlation (HTTP Requests, Sessions, Time) 03-24-2000
HTTP Requests
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

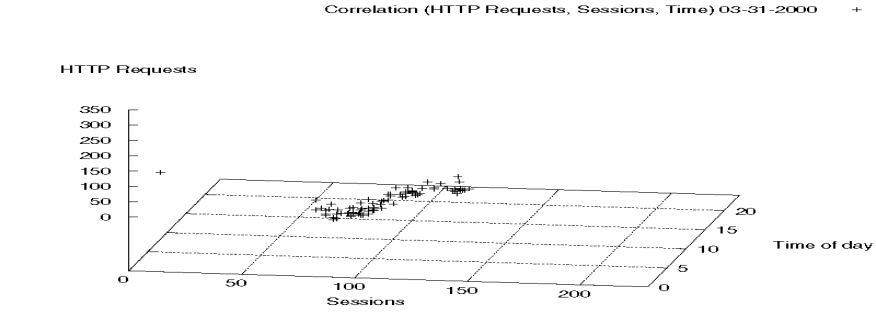

HTTP Requests
Correlation (HTTP Requests, Sessions, Time) 03-24-2000
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

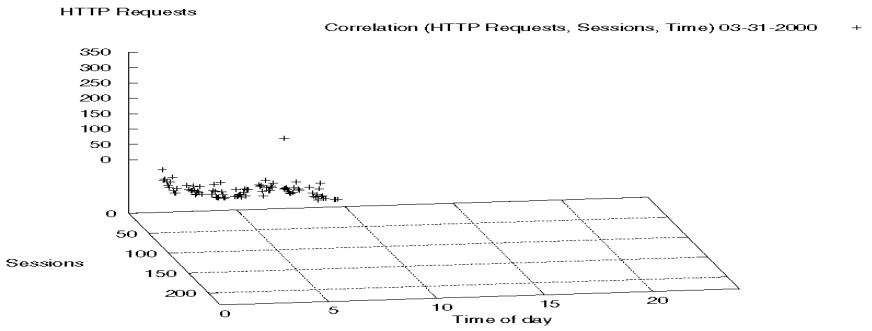

Correlation (HTTP Requests, Sessions, Time) 03-29-2000
HTTP Requests
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
20
15
10
5
0
Time of day

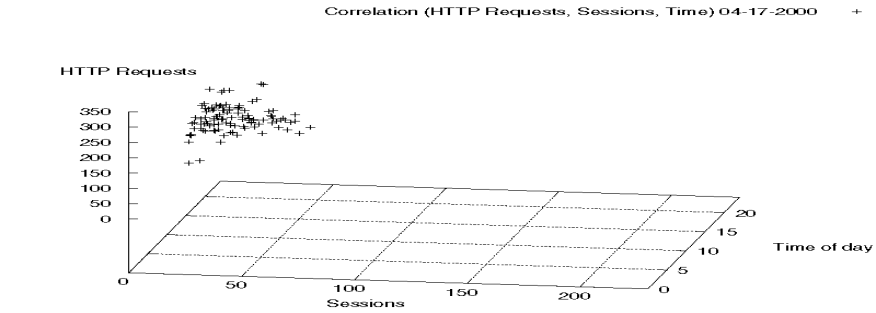

HTTP Requests
Correlation (HTTP Requests, Sessions, Time) 03-29-2000
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

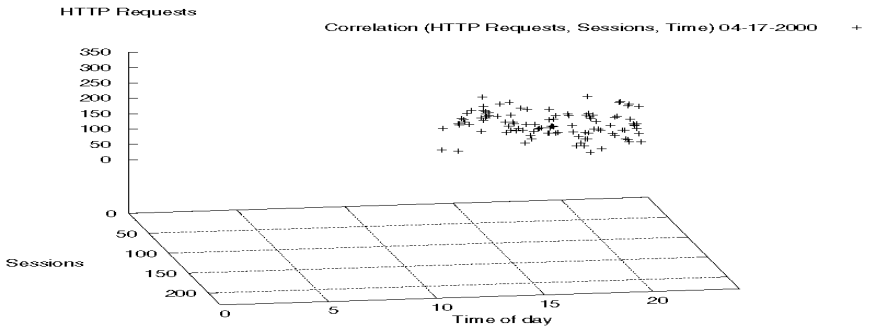

Correlation (HTTP Requests, Sessions, Time) 03-30-2000
HTTP Requests
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

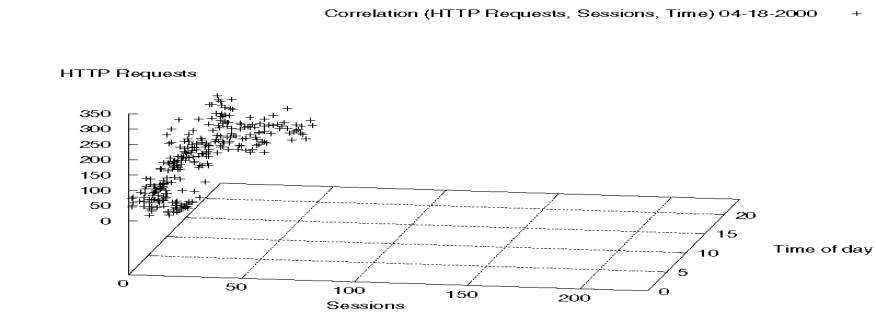

HTTP Requests
Correlation (HTTP Requests, Sessions, Time) 03-30-2000
350
300
250
200
150
100
50
0
Sessions
0
50
100
150
200
0
5
10
15
20
Time of day

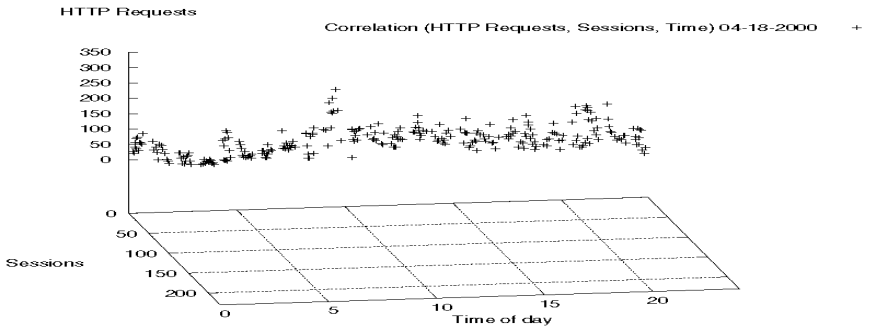

Correlation (HTTP Requests, Sessions, Time) 03-31-2000
HTTP Requests
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

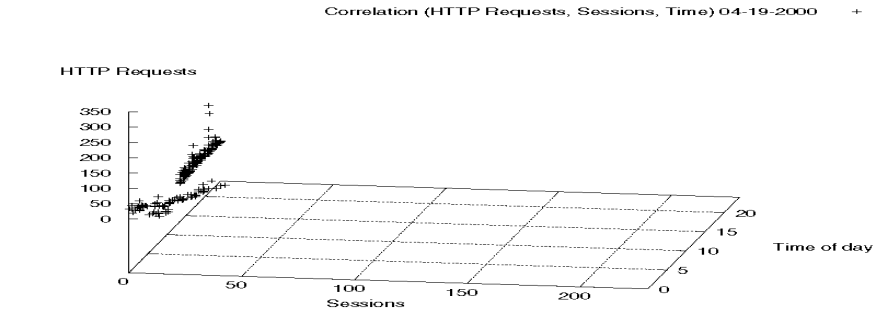

HTTP Requests
Correlation (HTTP Requests, Sessions, Time) 03-31-2000
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

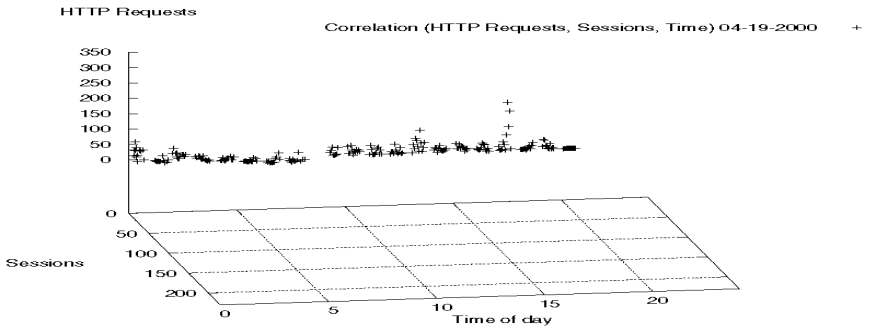

Correlation (HTTP Requests, Sessions, Time) 04-17-2000
HTTP Requests
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
20
15
10
5
0
Time of day

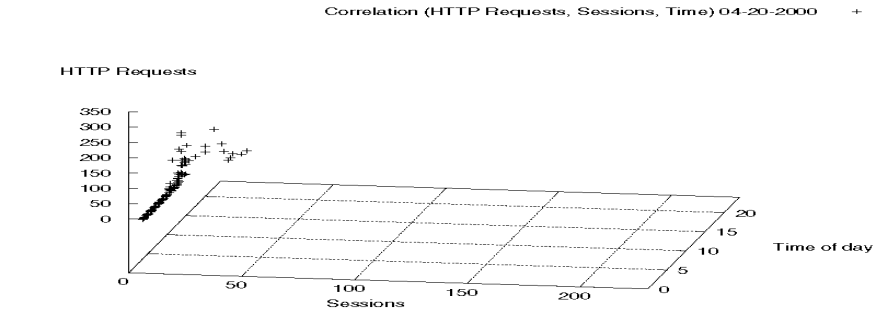

HTTP Requests
Correlation (HTTP Requests, Sessions, Time) 04-17-2000
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

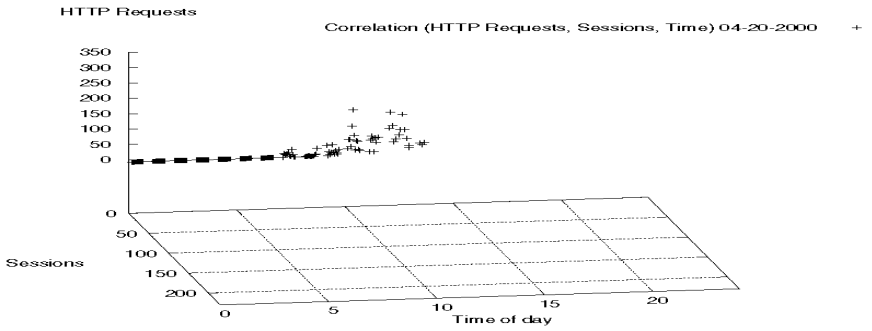

Correlation (HTTP Requests, Sessions, Time) 04-18-2000
HTTP Requests
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

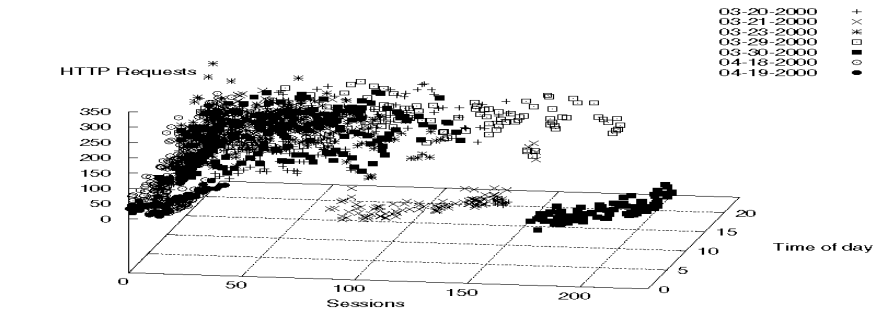

HTTP Requests
Correlation (HTTP Requests, Sessions, Time) 04-18-2000
350
300
250
200
150
100
50
0
Sessions
0
50
100
150
200
0
5
10
15
20
Time of day

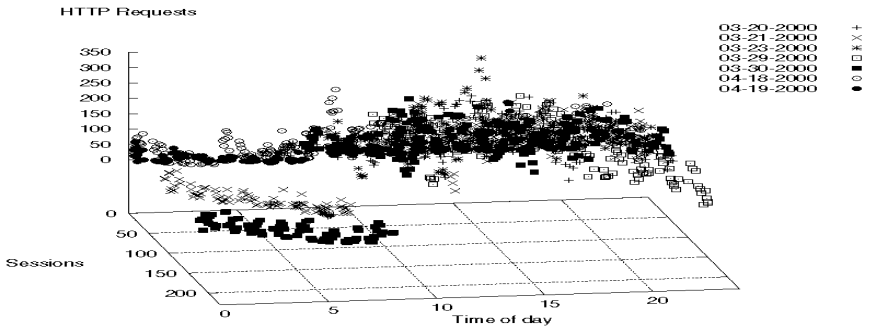

Correlation (HTTP Requests, Sessions, Time) 04-19-2000
HTTP Requests
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

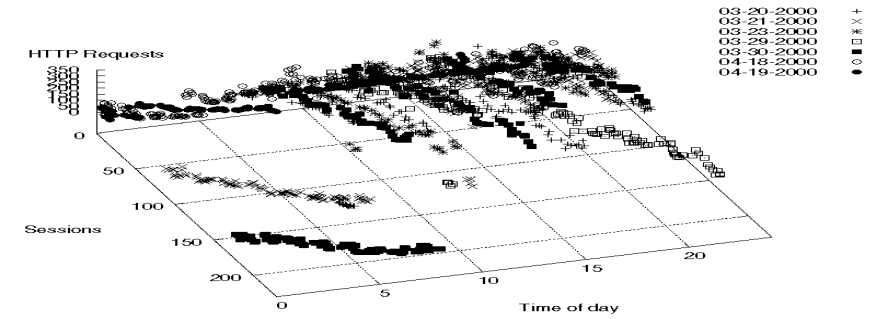

HTTP Requests
Correlation (HTTP Requests, Sessions, Time) 04-19-2000
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

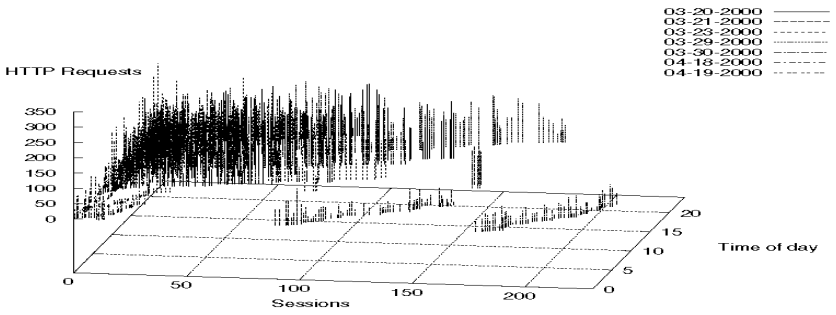

Correlation (HTTP Requests, Sessions, Time) 04-20-2000
HTTP Requests
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
20
15
10
5
0
Time of day

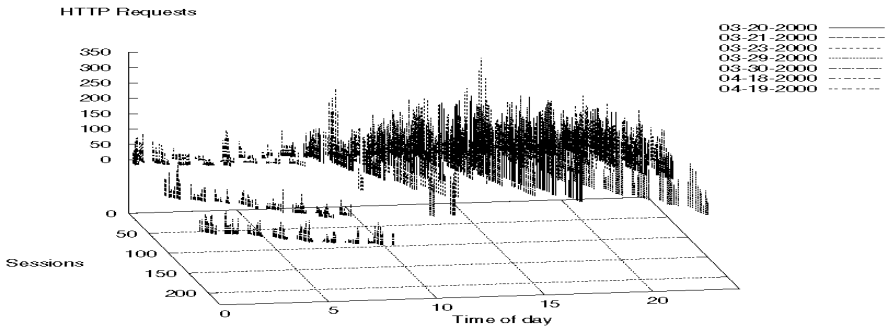

HTTP Requests
Correlation (HTTP Requests, Sessions, Time) 04-20-2000
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day

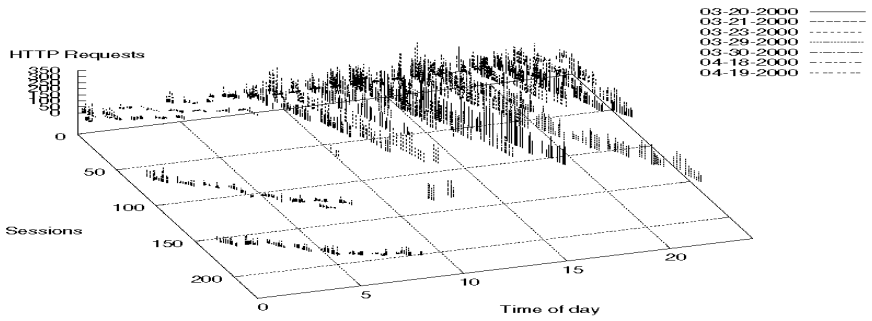

03-20-2000
03-21-2000
03-23-2000
03-29-2000
03-30-2000
04-18-2000
04-19-2000
HTTP Requests
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day


HTTP Requests
03-20-2000
03-21-2000
03-23-2000
03-29-2000
03-30-2000
04-18-2000
04-19-2000
350
300
250
200
150
100
50
0
Sessions
0
50
100
150
200
Time of day
0
5
10
15
20


HTTP Requests
03-20-2000
03-21-2000
03-23-2000
03-29-2000
03-30-2000
04-18-2000
04-19-2000
Sessions
0
50
100
150
200
Time of day
0
5
10
15
20


HTTP Requests
03-20-2000
03-21-2000
03-23-2000
03-29-2000
03-30-2000
04-18-2000
04-19-2000
350
300
250
200
150
100
50
0
0
50
100
150
200
Sessions
0
5
10
15
20
Time of day


HTTP Requests
03-20-2000
03-21-2000
03-23-2000
03-29-2000
03-30-2000
04-18-2000
04-19-2000
350
300
250
200
150
100
50
0
Sessions
0
50
100
150
200
0
5
10
15
20
Time of day


HTTP Requests
03-20-2000
03-21-2000
03-23-2000
03-29-2000
03-30-2000
04-18-2000
04-19-2000
Sessions
0
50
100
150
200
0
5
10
15
20
Time of day

E Tables

Appendix E presents the characteristics of the measured data we have made use of in the creation of the graphs shown in Appendix D. The days and time intervals are shown in the form of tables.

E.1 The Characteristics of Seven Measured Days

Date	**Duration**
03-20-2000	06.00 – 23.55
03-21-2000	00.00 – 07.40 12.00 – 23.55
03-23-2000	09.00 – 23.55
03-29-2000	11.01 – 23.56
03-30-2000	00.01 – 23.56
04-18-2000	00.03 – 23.57
04-19-2000	00.00 – 07.57 09.20 – 23.56

E.2 The Characteristics of Eight Measured Days

Date	Duration
03-20-2000	06.00 – 23.55
03-21-2000	00.00 – 07.40 12.00 – 23.55
03-23-2000	09.00 – 23.55
03-29-2000	11.01 – 23.56
03-30-2000	00.01 – 23.56
04-17-2000	14.44 – 23.58
04-18-2000	00.03 – 23.57
04-19-2000	00.00 – 07.57 09.20 – 23.56

E.3 The Characteristics of All Twelve Days

Date	Duration
03-20-2000	06.00 – 23.55
03-21-2000	00.00 – 07.40 12.00 – 23.55
03-22-2000	00.00 – 14.15
03-23-2000	09.00 – 23.55
03-24-2000	00.00 – 09.00
03-29-2000	11.01 – 23.56
03-30-2000	00.01 – 23.56
03-31-2000	00.01 – 07.21
04-17-2000	14.44 – 23.58
04-18-2000	00.03 – 23.57
04-19-2000	00.00 – 07.57 09.20 – 23.56
04-20-2000	00.01 – 13.10