Application Layer Tracing for Performance Evaluation

Ajit K. Jena

Computer Center
Indian Institute of Technology
Bombay-400076, India
ajit@cc.iitb.ac.in

Adrian Popescu, Arne A. Nilsson

Dept. of Telecommunications and Signal Processing
Blekinge Institute of Technology
371 79 Karlskrona, Sweden
apo@bth.se, ano@bth.se

Abstract

The paper reports on a set of non-intrusive tools that can be used for the performance evaluation of "classical" Internet applications such as Email, WWW, and FTP. At the application layer, key factors influencing the events are stochastic aspects of user behavior, protocol characteristics and sizes of contents transferred. A multi pronged approach involving application log analysis, direct probe of the WWW pages, and network flow analysis has been used. The measurement infrastructure is presented and diverse software tools for gathering of application layer properties are described in detail. The motivation is to build stochastic models for the key elements of application layer that can be further used to build up an integrated end-to-end performance testbed for conducting traffic engineering experiments.

1 Introduction

The global Internet has seen tremendous growth in terms of nodes and user base as well as of types of applications. An important consequence of this growth is related to an increased complexity of the traffic experienced in these networks. In order to improve the performance of the network, it is important to understand and to characterize the application level transactions as well as the effect of different TCP/IP control mechanisms on application-level parameters.

Internet applications like SMTP, HTTP and FTP have been in use for quite some time. Especially the popularity of WWW technology has had a major impact on modern life and has influenced our lives in many ways. It has, for instance, acted as a catalyst in the synergy of traditional telephony and data communication technologies. The Web technology has been quickly adapted for specific purposes like education,

entertainment, and commercial environments and has become the preferred way of content distribution. As a consequence, it is important to understand the complexity of WWW and to develop appropriate workload models for simulation and testbed purposes, in order to improve the performance. The problem is further complicated because the Web is actually consisting of a variety of different software components (e.g., browsers, servers, proxies, back-end databases), and also due to the ever-changing characteristics of Internet traffic.

The paper reports on several tools that can be used for performance evaluation of applications like Email and WWW. Due to the huge volume of data traffic collected, packet monitoring must be complemented by specific software to process the collected data, e.g., to summarize the huge information generated from the flows, to extract diverse application specific features, and to capture the message timings [6, 11].

A multi pronged approach involving application log analysis, direct probe of the WWW pages, and network flow monitoring and analysis has been used in our experiments. The objectives are to capture the type and sizes of application layer objects, structural similarities or differences among characteristics of sessions belonging to different types of applications as well as to search for possible invariant characteristics in object flows across different user organizations. The motivation is to build stochastic models for key elements of application layer that can be further used to build up an integrated end-to-end performance testbed for conducting traffic engineering experiments.

The rest of the paper is as follows. Section 2 gives a brief outline of the key elements at the application layer. Section 3 describes the measurement infrastructure used in our studies. Sections 4, 5, and 6 describe in detail how different specific methods have been used to summarize application flows. Section 7 describes a typical example of software, used to produce a flow summary to extract features of SMTP. The paper ends with a few general remarks about the experience gained during the development and use of these tools.

2 Application Characteristics

There are many different types of events that may occur at the application layer. These events are influenced by factors like the stochastic aspects of user behavior, protocol characteristics (controlling the sessions between client and server) as well as sizes of contents transferred (with heavy-tailed properties). These are factors that must be considered when sniffing traffic flows and deriving models for the application entities.

Client-server protocols used in Internet applications are based on a request-response messaging paradigm [2, 5, 7, 15]. The client usually sends command messages and the server responds to those commands. Often the client may not issue the next command until the present command is replied to by the server (e.g., in the case of SMTP), exhibiting so a kind of lock step behavior. In such cases the client goes into an OFF state after the command was issued. The duration of the OFF state is also dependent on the conditions prevailing inside the network. In the case of HTTP, the client side may invoke several commands in concurrently. These concurrent command messages may be transmitted using multiple TCP connections (e.g., HTTP 1.0) or in a pipelined

fashion on a single TCP connection (e.g., HTTP 1.1).

Typically, the actions at the application layer are triggered by users, within the frame of a session. During a session, one or more information entities (of user interest) are transferred over the network, and each of these transfers may be viewed as a transaction. A typical example is given in fig. 1 for WWW downloads. Here, the **main page** and the **embedded item** downloads constitute the transactions within a WWW session. Similar models can be also used for FTP and SMTP sessions, as reported in [9, 10].

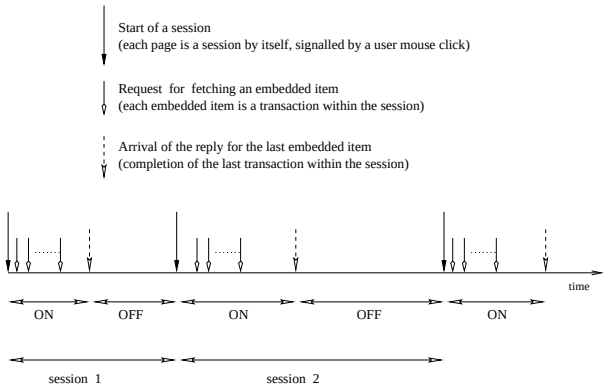

Figure 1: HTTP session

3 Measurement Infrastructure

In order to model the ON-OFF behavior of the application sources, it is essential to collect the protocol message elements along with their respective message timings. Measurements of application layer elements can be done in a variety of ways. For instance, in the case of WWW, diverse software elements (i.e., browsers, proxies and servers) may generate logs as part of the normal operation (of handling requests). While these methods have the advantage of simplicity, the price however is that of severe limitations in the detail of information that can be logged, with the consequence that they are not necessarily representative of the overall Web. De facto standards available today for proxy and server logging are the Common Log Format (CLF) and the Extended Common Log Format (ECLF) [11].

Logging information should therefore be complemented by packet monitoring and, eventually, active measurements, from the wire. Web mirroring software can also provide useful information. The advantage in this case is that the information gathered (from packet monitoring) includes full application header information as well as detailed information regarding protocols below the application. By this, diverse protocol interactions, e.g., between HTTP and TCP, can also be captured, with the consequence of better performance evaluation. The drawback however is that packet monitoring is applied in an informal manner, and the output format may vary from one implementation to another.

One of the main bottlenecks in the process of collection of application layer information is the issue of protecting the privacy of the users. This is especially valid in the case of FTP and Email transfers, which are rather personal and sensitive. In the case of WWW, the content may be public but profiles of individual user access is quite sensitive. Due to the sensitive nature of information content, it is often difficult to monitor the data, which is needed for modeling. Many different approaches are available today for collecting application layer information, and various organizations may even enforce different mechanisms to regulate monitoring of data traffic in own networks [11].

In order to characterize Internet applications under the uniform framework of transaction-oriented applications, a combination of several methodologies has been adopted in our case, namely application logging, direct probing as well as monitoring of network data flows [9, 10]. Accordingly, several measurement utilities have been used for doing traffic monitoring and analysis of data at both client and server sides:

- The NIKSUN NetVCR^{TM} monitoring and analysis system [12] is a non-intrusive system for network monitoring and analysis that is able to collect and storage data from different link equipments. The system can be connected to different kinds of interfaces in a specific local/wide area network (LAN/WAN) and collect all data traffic in the specific LAN or on the probed WAN interfaces. The observed data traffic can be recorded for later reference on a storage device such as hard disk or tape. Individual packets can be automatically classified into link to application layer data sets and automatically analyzed (e.g., Frame Relay, Ethernet, ATM, IP, TCP, UDP, WWW, MBone). NetVCR's Application Programming Interface (API) allows for retrieving of stored data traffic based on some criteria (e.g., time interval, packet type, IP address, port number).

- The ***tcptrace*** utility, for mapping of IP packets to TCP flows [13], is used to increase the capabilities of NetVCR^{TM}.

- The ***augmented tcptrace*** utility is used to capture the message timings [9].

- The ***x_flowstat*** utilities, where the term ***x*** refers to any of the three applications (i.e., ***smtp*** or ***http*** or ***ftp***), is used to help summarize the huge information generated from the flows and to extract diverse application specific features [9].

- ***GetWeb*** mirroring software is used for duplicating the content tree of a designated Web server on the local disks [17], and

- Using of application logs analysis (especially SMTP logs and access logs at FTP servers).

The objectives for the monitoring and analysis infrastructure are to capture the following:

- Type and sizes of application layer objects, for building up session structure models for the three applications.

- Structural similarities and/or differences among characteristics of sessions belonging to different types of applications, and
- Search for possible invariant characteristics in object flows across different user organizations.

Extensive studies of performance analysis have been done for SMTP, HTTP and FTP, on both client and server side, and with traffic collected from different places in the world. These studies are reported in [9, 10].

4 Application Logs

The simplest method of collecting application layer information is in the form of application logs. Such logs are quite useful in many cases. However, under the present session-transaction oriented structure, the scope of application logs is rather limited. In the following, the efficacies of application logs for the present modeling work are discussed for each of the applications considered:

- For Email, two popular mail server processing software are **sendmail** and **qmail** [14]. The logs produced by these can be processed to produce one line for each mail, providing so information about sender, recipient, and the message size. In the specific case of our project, logs at the Blekinge Institute of Technology, Karlskrona, Sweden and the Indian Institute of Technology, Bombay, India were processed. In both cases, only the mails actually transacted over the Internet were considered and the intra-campus mails were dropped. This method is effective in collecting the mail message sizes but it does not reveal multi-transaction Email sessions. As a result, the distribution of the number of mail message transfers within a single session was not inferred. Thus, in order to build a better application level model, this has to be augmented with analysis of live SMTP flows from the network.
- Certain FTP servers permit logging of user activities as a runtime option. For each FTP session, the user login and logout events and each of the transfers (upload or download) together with the number of bytes transferred can be recorded. Using such information, FTP sessions can be characterized in terms of the number of bytes transferred during transaction, number of transactions in a session, GET vs PUT probability of a transaction, etc.
- In the case of WWW, access logs may be maintained both on the client and on the server side. A single line per request is recorded by most industry standard servers in a special format specified by W3C working group [16]. This is useful for drawing inferences about the size of Web objects. The client side logs are mostly associated with the NCSA Mosaic browser. Most modern browsers in the post Mosaic era have dicontinued the practice of generating client access logs. Just like the server side logs, the client side logs can also be used to determine Web object sizes that are downloaded. It can be used to determine the user preference of one server over others. In some research, WWW client side logs have been

used to model the number of embedded items within a mainpage [1]. WWW client logs make no distinction between mainpage or embedded type accesses. So, such research is based on the principles of clustering. This is based on the hypothesis that all accesses between a host pair within a time window of about 15 milliseconds belong to the same Webpage. However, such hypothesis could be inaccurate because of certain reasons [9].

5 Direct Probe Method

In addition to the regular browser based interface of accessing the WWW information, occasionally the Web can be accessed via Web mirroring software (sometimes also known as **robots**). The primary purpose of this software is to mirror the entire content tree of a designated server. This method is often adopted to create proxy contents in order to improve the performance. Given a URI, this software first downloads the specific URI. It then analyzes the downloaded contents and finds all the embedded items within the specific page. All the embedded items are then fetched. At this stage, the given URI is complete. The software then further analyzes the hyperlink references located within the downloaded page. Each of the hyperlinks are downloaded one after another. Thus, the software recursively replicates the WWW tree structure resident under the user specified URI. The level of recursive descent is a user specified parameter.

The transactions within WWW sessions are created by user actions and represent the combined effect of two physical processes. First, the user selects a particular server out on the Web and then he may choose a particular subset of the contents of the specific server. For example, if the server is a news media related then, on any given day, the maximum traffic will be naturally directed towards the particular day's news pages. All other pages will be resident on the server but may not contribute substantially towards the transfer characteristics of the specific day's traffic.

WWW is a server-centric application in that it generates most of data flow from the server towards the clients. The server has ***latent*** properties in terms of the content pages it holds as well as the structural properties of these pages. The structural properties refer to the number of *embedded items* in a page, the types of *embedded items*, etc. Some of the ***latent*** properties of a server are exhibited during WWW sessions, when the user first selects a specific server and then from that server he chooses a portion out of the total domain on WWW pages. In terms of analogy, the situation is similar to the energy systems where the objects, by virtue of their location, have potential energy. Some of the potential energy gets converted into kinetic energy when the object moves. One of the main goals of our experiments was to study both the ***latent*** and the ***exhibited*** properties in the WWW application. The ***latent*** properties are captured using the ***GetWeb*** software whereas the ***exhibited*** properties can be collected by monitoring WWW application flows from the network.

For the purposes of modeling work, public domain mirroring software known as the **GetWeb** [17] has been used. The software is targeted towards downloading Web pages and restructuring them so that the downloaded pages can be browsed offline. The software has been modified in the sense that it downloads the content pages, analyzes

the contents and reports the content types and the sizes [9]. It also counts the number of embedded items and classifies the embedded references as **external** (i.e., resident on a host different from the host holding the mainpage) or **internal** (i.e., embedded item co-located with the mainpage on the designated server).

Using this software a large cross section of the WWW servers have been probed. The set of servers were first carefully chosen so as to cover different purposes the Web is used for by the user community, namely, entertainment, education, and commercial. The motivation for the direct probing is actually twofold. The main purpose is to build a session structure model of the WWW transfers. The other goal is to get complete information about the types and sizes of the Web pages resident at different servers. A clear distinction has to be made between the objects resident on a server and the objects actually transferred and hence appearing in flow monitor dumps. The first is a property of the server itself whereas the second reflects a cross section of the entire object space that the users actually access. Both properties are relevant in the characterization of WWW service.

6 Monitoring Network Flows

Monitoring packet flows on the wire is a powerful method of collecting application layer flows. Information objects transferred during the sessions of distributed applications are mapped into packet flows. As a result, both session and transaction properties can be collected. This section describes implementation details of our method and how this has been used to collect different properties related to different applications.

An important component is the packet sniffer engine. The basic criteria to be met by this engine is the ability to continuously gather packet traces from high speed medium and, secondly, to provide easy access to the gathered traces. To this end, a commercial hardware and software called NetVCR has been used [12]. The platform provides access to the collected traces using a number of search criteria such as a designated time duration, a host pair, source or destination port, etc. The packet sniffing software is based on **tcpdump** [8], which copies a specified number of bytes from the specified network interfaces. In addition, a proprietary indexing mechanism is used to store the recorded packets. The indexing uses a hashing algorithm based on the IP header fields so as to identify the packet uniquely. In addition to the flexibility of storage and retrieval of traces, this mechanism allows for computing one way end-to-end packet delays. This is because the packet at the source end can be exactly tallied with the packet at the sink end.

The traces obtained from the sniffing platform, by themself, can not meet the requirements of application layer tracing. Substantial post processing is required to achieve this goal [6, 9]. Such post processing involves two aspects, namely mapping packet traces onto TCP flows, and processing the resultant flows using application specific software that extracts the relevant attributes.

In order to map packet traces onto TCP flows, we use a public domain software called **tcptrace** [13]. Each connection between two end systems over the Internet creates two flows: one from the initiator to the responder and the other in the reverse direction. Given a packet trace, the software strips off the link layer and the IP header

information from the packet. Subsequently, the TCP window field (in the TCP header), along with TCP flags (SYN, FIN, and PSH) are analyzed. The packet flow is also analyzed to handle TCP retransmissions. After the TCP headers are processed, we obtain the demultiplexed application layer data that is further passed to the process handling the application protocol in a real application situation. To facilitate collection of session arrival timings and application layer protocol message timings the basic capabilities of the **tcptrace** software have been augmented to collect packet arrival timings. It is known that application layer messages may often span across multiple packets. In such cases, the first and the last packet containing the message are significant.

The processing of TCP flows for the application layer events of the individual application types depends on the specific application layer protocols and details of their implementation. As a result, the processing is different for each application. Details of the tracing software for SMTP are provided in section 7. Details of other tracing software (for HTTP and FTP) are provided in [9].

7 The *smtp_flowstat* Software

The **tcptrace** software outputs the application level data exchanges (and the associated timings) into separate files. For each TCP connection, two files are produced, one for each half of the connection. It is mentioned that, for the purposes of the current work, the timings of the application layer messages are considered to be the same as the timings of the link layer packets.

In order to understand the ***smtp_flowstat*** software, we consider the protocol exchanges during an Email session (figure 2). The software tool consolidates the two halves of the application layer dialog (that occurs during an Email connection), and produces one summary file for each. Figure 3 shows the format for a typical SMTP session with the summary of both protocol exchanges and timings.

The SMTP session log shown in figure 3 can be regarded as a collection of records with each line being a record. Each record has several blank separated fields.

The first field of the records represents the record type. It may be CMD, PKT, TXN, or SESS. The type CMD indicates that the record is a command from the SMTP initiator to its peer on the other side. The type PKT indicates that it is simply a data packet being transmitted from the initiator side to the responder side. Such records will be created only during the transmission of SMTP message body. Finally, the TXN and SESS types are summary records showing the transaction and session details of the particular SMTP exchange under consideration.

The second field of the CMD and PKT type records contains the real timestamp that was recorded by the NetVCR monitor when the packet containing the message was observed on the link. For example, the first record in figure 3 shows a timestamp value of 931199433.153363. Actually, this is the UNIX epoch value used by the NetVCR monitor as a timestamp. This is the absolute timestamp when the TCP connection request originated from the SMTP initiator. Similarly, all the subsequent CMD type records show the timestamp values when the respective SMTP messages were recorded. The second field for TXN and SESS record types contain the connection identifier. The identifier is based on the real time when the connection was initiated concatenated with

Figure 2: Message exchanges during an SMTP session

```
CMD 931199433.153363 0.000000 0 0 INIT RESP 931199433.208986 55.623055 13894 85
CMD 931199433.215003 61.640024 13895 19 EHLO RESP 931199433.215643 62.280059 13897 175
CMD 931199433.220398 67.034960 13898 49 MAIL RESP 931199433.229533 76.169968 13900 44
CMD 931199433.232750 79.387069 13901 39 RCPT RESP 931199433.239515 86.151958 13903 51
CMD 931199433.242945 89.581966 13904 6 DATA RESP 931199433.243423 90.059996 13905 50
PKT 931199433.265146 111.783028 13906 1460 DPKT
PKT 931199433.266382 113.018990 13907 1460 DPKT
PKT 931199433.267613 114.250064 13908 1460 DPKT
PKT 931199433.269003 115.640044 13910 1460 DPKT
.
.
.
.

PKT 931199435.219443 2066.079974 14119 1460 DPKT
PKT 931199435.220747 2067.384005 14121 1460 DPKT
PKT 931199435.222137 2068.773985 14122 1460 DPKT
PKT 931199435.224284 2070.921063 14124 1460 DPKT
PKT 931199435.225059 2071.696043 14125 892 DPKT
CMD 931199435.225830 2072.466969 14126 203831 DATA_END RESP 931199435.394059 2240.695953 14129 44
CMD 931199435.449388 2296.025038 14131 6 QUIT RESP 931199435.449829 2296.465993 14132 45
TXN 931199433.153363.cga2cgb 0 1 203831

SESS 931199433.153363.cga2cgb 1
```

Figure 3: Typical SMTP session summary

the flow identifier assigned by the ***tcptrace*** utility described earlier.

The third field of CMD and PKT type records shows the relative time (in milliseconds), with respect to the connection start epoch. The first record corresponds to the TCP connection initiation from the initiator to the responder. As a result the field for the first record will always be zero. For all other relevant record types this will indicate the relative protocol message time (in milliseconds) or the data packet time. For example, the third field of the second record shows a value of 61.640024 milliseconds. This is the elapsed time since the first record. For the TXN type records, the third field contains the transaction *id* belonging to the session. The transaction *id* is given by a counter (starting with zero), which is assigned to identify a transaction. Consequently, only multi-transaction email sessions will have multiple TXN records and this field will be used to identify each transaction uniquely. Finally, for the SESS type records, the third field indicates the total number of transactions seen during the session.

The fourth field in CMD, PKT, and TXN records (SESS records have only three fields) indicates the packet number. This number is an unique number assigned by the NetVCR dataset creation utility. The packet numbers are strictly increasing sequence numbers assigned across all flows. As a result, the packet numbers belonging to a flow may not be continuous but they will form an increasing sequence. The first record corresponds to a connection request packet seen at the TCP level. As a result, this contains no application layer information and as such the packet number is shown as zero. This first record is a synthetically created one and serves the purpose of a time base. All other timings are with respect to this record. The fourth field in the TXN record shows the total number of recipients for the particular transaction under consideration. In essence this field carries the total number of recipients (To: Cc: and Bcc: fields in the mail header) for the mail message. For example, the record for TXN in figure 3 shows the third field value as 1 indicative of the fact that the particular mail transaction has only one recipient.

The fifth field for CMD records shows the message size. Since the SMTP protocol messages are pretty small (less than 200 bytes [14]), each protocol message usually appears as a single packet. As an example, the third record has the fifth field value 49 indicating that the protocol message under consideration was 49 bytes long. For the PKT record the fifth field shows the length of the data packet transmitted. The data packets are created only when the session is in a transaction mode. A SMTP session is in transaction mode only when a message body is being actively transmitted. At all other times it is either preparing to enter a transaction mode or coming out of a transaction mode. The NetVCR monitor used for our work has only Ethernet interfaces. Thus bigger application level messages are seen as fragmented Ethernet packets carrying up to a maximum of 1460 payload bytes. In addition, the implementation of the application may further use application layer buffers. During the analysis phase, it was observed that most implementations of SMTP actually transfer the mail message bodies in chunks of 8192 bytes. This may occur due to the buffering at application layer or even due to the windowing mechanism of TCP. The cyclic patterns of five 1460 byte packets followed by one 892 byte packets are noticed. The operating systems involved in the mail exchanges are different flavors of UNIX and to a lesser extent WINDOWS. In the flowstat summary, the first record of type PKT is a synthetic one and as such it has been assigned a message size of zero.

The sixth field in the PKT and CMD type records contains the command type. This encapsulates the protocol elements shown in the initiator half of the figure 2. There is no protocol element INIT in SMTP exchange dialogues. However, this is used in the first record to highlight the epoch where all the activity starts. Other command types are HELO (or EHLO), MAIL, RCPT, RSET, DATA, QUIT, etc. The RSET type is used in multi-transaction SMTP sessions before the beginning of each transaction. The type DATA is used when the actual message body is transmitted from the initiator to the responder. The PKT record types are really not application layer messages. They have been given command type DPKT just to identify them as belonging to a message body. Similarly, the command type DATA_END really does not exist. This has been added to signal the end of a message body transmission (this is specified as a '.' on a single line by itself in the SMTP standards).

The fields greater than six describe the response messages from the responder side. Thus, they appear only for CMD type records. The ***smtp_flowstat*** tool pairs up the initiator command messages with the corresponding response message. It is pointed out that the CMD record of INIT type is paired with the initial identification response from the responder side.

The seventh field, wherever it is present, will always be RESP. This indicates that the fields following after this will be treated as the response from the SMTP responder. As a consequence, this field will appear only for CMD type records.

The eighth field indicates the epoch time of the response message. This field has semantics similar to that of the second field in CMD type records.

The ninth field shows the relative time (following the same semantics as for field three) for the response message. For example, the second record shows a value of 62.280059. This is the time in milliseconds when the EHLO response message was recorded by the NetVCR monitor.

The tenth field shows the packet number (similar to the fourth field). Finally, the last field shows the size of the response message.

In summary, the ***smtp_flowstat*** tool produces a flow summary that provides a flexible mechanism to extract SMTP application features. It encapsulates complete information about the SMTP flows, which can be further used for modeling purposes.

8 Summary

The paper describes various methodologies applicable in gathering stochastic properties at the application layer, and their implementations. Classical Internet applications such as Email, FTP, and Email are modeled using a uniform framework. Under this framework, each service access by the user is seen as a session wherein multiple transactions are executed. Each transaction amounts to the transfer of an information entity such as an Email message, or a file.

Using the software tools described above, statistics such as user session arrivals, protocol message sizes, and the number of transactions likely to occur within a session, can be obtained. These tools form part of a simulation testbed to study the impact of the stochastic aspects of application object sizes on LRD properties at link layer and to evaluate quality of service issues as a function of TCP and link layer controls.

References

[1] Barford P. and Crovella M., *Generating Representative Web Workloads for Network and Server Performance Evaluation*, ACM Sigmetrics, 1998.

[2] Berners-Lee R. et al., *Hytertext Transfer Protocol – HTTP/1.0*, RFC 1945, May 1996.

[3] Borenstein N. and Freed N., *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, RFC 2045, November 1996.

[4] Borenstein N. and Freed N., *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, RFC 2046, November 1996.

[5] Crocker D.H., *Standard for the Format of ARPA Internet Text Messages*, RFC 822, August 1982.

[6] Feldmann A., *BLT: Bi-Layer Tracing of HTTP and TCP/IP*, World Wide Web Conference, May 2000.

[7] Fielding R. et al., *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616, June 1999.

[8] Jacobson V et al, *tcpdump software*, ftp://ee.lbl.gov, June 1989.

[9] Jena A.K., *Modeling and Analysis of Internet Applications*, Licentiate Thesis, Lund Institute of Technology, Sweden, June 2000.

[10] Jena A.K., Popescu, A. and Nilsson, A.A., *Modeling and Evaluation of Internet Applications*, submitted to the IFIP WG 7.3 International Symposium on Computer Performance Modeling, Measurement and Evaluation, Rome, Italy, September 2002.

[11] Krishnamurthy B. and Rexford J., *Web Protocols and Practice, HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement*, Addison-Wesley, 2001.

[12] NIKSUN NetVCR, http://niksun.com/products/netvcr.html

[13] Ostermann S., *TCPTRACE: A TCP connection analysis tool* , http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html

[14] Postel J.B., *Simple Mail Transfer Protocol*, RFC 821, August 1982.

[15] Postel J.B. and Reynolds J.K., *File Transfer Protocol*, RFC 959, October 1985.

[16] W3C group, *Logging Control in W3C httpd* , http://www.w3.org/Daemon/User/Config/Logging.html

[17] Xanthakis S., *GetWeb version 2.7.2*, http://www.enfin.com/getweb