Modeling of Internet Applications

Ajit K Jena and Adrian Popescu

Department of Telecommunications and Signal Processing,
University of Karlskrona/Ronneby,
371 79 Karlskrona, Sweden.
E-mail: akj@its.hk-r.se, adrian@itm.hk-r.se

Abstract

The global Internet has seen tremendous growth in terms of nodes and user base as well as of types of applications. One of the most important consequences of this growth is related to an increased complexity of the traffic experienced in these networks. Each application has a set of unique characteristics in terms of performance characteristics, transactions as well as the way the transaction processing profile maps onto unique network resource requirements. In order to support Internet applications effectively, it is therefore 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. It is the goal of this paper to model and to evaluate the characteristics of World Wide Web traffic. Results are reported on measuring, modeling and analysis of specific Hyper Text Transfer Protocol traffic collected from different (classes of) sites together with methodologies used for capturing HTTP flows as well as for modeling. The paper concludes with a discussion on the structure of Web pages and a model for the generation of the number of embedded pages in a Web page is suggested.

1 Introduction

The popularity of WWW technology has had a major impact on modern life and influenced our lives in many ways. It has, for instance, acted as a catalyst in the synergy of traditional telephony and data communication technologies. Furthermore, the fact that the WWW technology is available *on demand* [16] made it very appealing to the user community. As a result, the technology has been quickly adapted for specific purposes like education, entertainment, and commercial environments. In fact, it has dwarfed many other aspects of Internet, as in the case of FTP, which has almost been pushed out of fashion. The Web has become the preferred way of content distribution. As a consequence, researchers have been quite active and tried to improve the performance of WWW. Significant contributions have been made so far towards characterizing the application [2, 3, 5], towards better protocols [10, 18, 23] as well as for better caching strategies for clients and for servers [1]. Major challenges are scalability,

latency, bandwidth and the problem of aborted connections [9]. As a general observation it is mentioned that fundamental to all these studies is a deep understanding of WWW workloads.

The purpose of the paper is to present a characterization study of WWW clients and servers. Detailed results are reported on measuring, modeling and analysis of HTTP traffic collected from different (classes of) sites together with methodologies used for capturing HTTP flows as well as for modeling. In particular, we show that the distribution of document sizes can be modeled by a mixture of Lognormal and Pareto distributions as well as that diverse classes of Web servers (e.g., academic, entertainment, commercial) show structural similarities in their distributional properties. Furthermore, it is also shown that the number of embedded documents in a Web page can be well modeled by a Negative Binomial distribution. In addition, a model is put forth for the generation of the number of embedded pages in a Web page.

The paper is organized as follows. Section 2 is devoted to the HTTP session structure and the stochastic marked point process used to model the HTTP timings. Section 3 describes the measurement infrastructure used for this study. The methodology for capturing HTTP flows is presented in sections 4 and 5. Section 6 describes the HTTP characteristics that are considered in the study. Section 8 reports the summary statistics for the collected data sets together with the appropriate modeling methodology used (section 7). Finally, the paper concludes with a discussion on the structure of Web pages and a model is put forth for the generation of the number of embedded documents (section 9).

2 Session Structure

The WWW application is accessed typically through graphical Web browsers. The human user initiates the transfer of information from a server by clicking on links in a Web document. A Web page consists of objects. These (content) objects are transferred from the WWW server to the client. On the server, each of the objects is physically stored as a distinct file such as HTML file, GIF or JPEG image, audio clip, etc. The collection of all severs on the global Internet creates an information Web. On this Web, each object is associated with an unique address. The logical addressing scheme of the Web objects is defined using a mechanism called Uniform Resource Identifier (URI) [4].

The HTTP application layer protocol [4, 8, 14] is at the core of the WWW information network. The protocol uses the reliable TCP as the transport mechanism. Characterization of the WWW services essentially involves characterization of the objects transferred using HTTP. Further, another important issue is the way TCP connections are managed at the application layer. The Web means different things to different people. One server may be a newspaper type, a second one may be an university type, and the third one may be a site geared to provide entertainment. The content types may be different in each case. Thus, in order to characterize the Web objects, it is important to analyze the objects resident at a wide cross section of Web servers. Furthermore, another important issue is the user preference. The network traffic and host processing load at a server is mainly determined by the number of concurrent users. This is a highly subjective issue and thus difficult to characterize.

HTTP is a stateless protocol. This is in contrast to SMTP or FTP where both the client and the server maintain a lot of state information about the progress of sessions. A HTTP session starts when the client sends a request to the server for a specific document. The server processes the client requests without saving any context information about the clients. The server uses

the TCP connection opened by the client to send back the results. It is also not aware of the structuring of the Web pages. It is only the browser on the client side that interprets the fetched pages to find out the structure so as to determine the representation on the browser screen.

A stochastic marked point process is used to model the HTTP timings (fig. 1). The arrival times for the requests correspond to the beginning of sessions. Each session is associated with the following random variables:

- ON time: time duration elapsed for the fetching of the main Web page as well as the associated embedded items;
- active OFF time: time duration between consecutive fetches of embedded documents (within same ON period);
- passive OFF time: time duration elapsed between the completion of the transfer of one page and the beginning of the following session (user think time).

For the purpose of explaining the data collection and modeling aspects, a HTTP session is defined to start at the instant the user asks for an URI by explicitly naming it or by pointing the mouse at a link reference and clicking on it. With this epoch, the client side connects to the server and starts sending the request for the *main page* and subsequently the *embedded pages*. The server processes the requests and sends the contents back to the client. The session is said to have ended when the last embedded page is received at the client side. At this point, the view on the browser screen is complete and the human user starts reading the page. From the traffic point of view this is a silent period. We define this duration as the inter-session gap (i.e., passive OFF time). The inter-session gap represents the human user behavior (user think time). This is the actual time duration spent by the user with the page and typical action may simply involve browsing, printing, or even saving it on local disk for future reference.

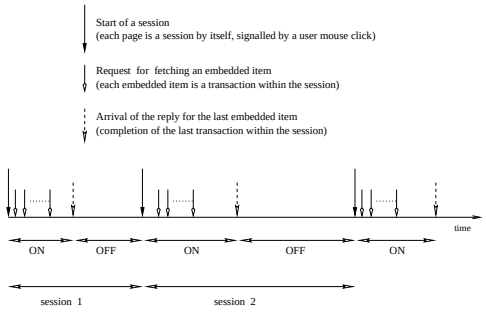

Figure 1: HTTP session

There are three types of HTTP requests:

- GET request: this is a method to fetch the information contained in the meta object specified by the requested URI;
- HEAD request: that is similar in intend but the purpose is limited to fetching the header information of the designated URI only. The aim is to test a hypertext link for validity, accessibility, and latest modification status;

- POST request: this is used to facilitate a reverse flow of information content from the client to the server. Here, the client is the generator of the information content that is intended to be processed and stored at the server. This is typically used for posting emails, news, or sending forms (e.g., opinion polls, Internet shopping, etc).

Fig. 2 shows typical formats used for the HTTP GET request and the corresponding response. The header fields have important roles to play during HTTP protocol exchanges. The headers in the HTTP requests have fields to indicate the capabilities of the browser so that the server can choose the appropriate version of the object. The header also contains information about the current version object (if it has been fetched in the past) specified via the *If-modified-since* field. Finally, the request header field *Connection* contains information about the HTTP protocol version and its specific options. The version and the option settings are used by the server for keeping the transport connection open or close it after each object. The headers in the HTTP responses identify the content type, contain information about its length, and the encoding methods so that the browser can apply the appropriate interpretation rules. The response header contains also the response code. One important use of the response code is to signal to the browser to fetch the object from an alternative URI. The response header may also contain the *Last-Modified* field so that the browser may update its in-cache copy. The *Connection* field in the response header indicates the persistent HTTP capabilities of the server. The persistence capabilities of the client and the server have to be compatible. The advanced features of HTTP become activated only under these circumstances. Otherwise, the transfers fall back to the default non-persistent HTTP mode.

Most Web pages are designed around an object called *base page* or *main page*. The *main page* serves as an anchor and holds, in its body, several references to other URIs. These references may be *links* or *embedded items*. The *links* are undelined or marked with special colors or symbols when they are rendered on the user screens. These are hints to the human user so that the user can further click on them and fetch them. On the other hand, the participating URIs corresponding to the *embedded items* are implicitly fetched and rendered on the user screen. Typically the *embedded items* are of type images, maps, frame layouts, audio clips, etc. Very often, the embedded references are located on the same server as the *main page* but in a few cases they reside on a different server and have to be fetched from there. The body of a typical *main page* is shown in figure 3. The HTTP body immediately follows the response header in the HTTP reply with a blank line separating the header and the body.

The body of the server response contains text mixed with tagged objects to identify HTTP items. Some of the HTTP items are hyperlinks. Typically these are the items that are rendered underlined or highlighted items so that the user can further click on to download. Such items are not embedded items. They require explicit user action to initiate their transfers from the respective servers. In addition, there are HTTP elements that do not require any user action and are implicitly fetched by the browser automatically without any explicit action on the user part. Fig. 3 illustrates some of the typical tagged items together with their respective categories extracted from actual Webpages. Notice the further classification of the embedded items as ”Internal” and ”External”. There are other types such as CODE, BODY, BGSOUND etc, which also represent embedded items.

```
GET /index.html HTTP/1.0
If-Modified-Since: Monday, 26-Apr-99 13:44:45 GMT ; length=26768
Connection: Keep-Alive
User-Agent: Mozilla/4.04 [en] (X11; I; SunOS 5.5.1 sun4u)
Host: www.hk-r.se
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

```
HTTP/1.1 200 OK
Date: Mon, 26 Apr 1999 15:56:54 GMT
Server: Apache/1.3.4 (Unix)
Last-Modified: Mon, 26 Apr 1999 15:48:00 GMT
Accept-Ranges: bytes
Content-Length: 28691
Keep-Alive: timeout=15, max=1000
Connection: Keep-Alive
Content-Type: text/html
```

Figure 2: Contents of a typical GET request header (top) and the corresponding response (bottom)

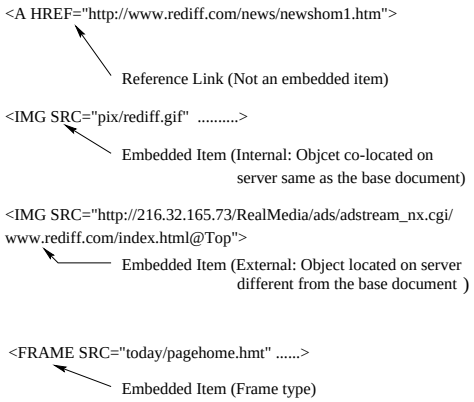

Figure 3: Content of a typical HTTP main page

3 Measuring Infrastructure

The NIKSUN NetVCR^{TM} monitoring and analysis system has been used for traffic measurements. NetVCR [17] is a non-intrusive system for network monitoring and analysis able to collect and storage data from link layer up to application layer. The system can be connected to all the interfaces in a specific local/wide area network (LAN/WAN) and collect all data traffic in that 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 are automatically classified into link to application layer sets and automatically analyzed (e.g., Frame Relay, Ethernet, ATM, IP, TCP, UDP, WWW, MBone). NetVCR's application programming interface (API) allows retrieving of stored data traffic based on some criteria (e.g., time interval, packet type, IP address, port number, etc).

We used NIKSUN NetVCR system and the Direct Probe (as described in section 5) to collect data from a number of sites (as described in section 8). The content pages on WWW servers are created by the hosting organizations towards the specific goal of furthering their interests. To that extent, the pages represent the characteristics of the organization. We hypothesized that organizations with similar business ethos have similar content structures and should therefore exhibit some similarities in structure. Accordingly, three classes of servers were identified as possible targets, namely educational, media companies, and companies engaged in information technology business. In each of these categories, three to four servers were probed. Starting at their homepages, and using a recursive technique, we descended 10 levels of the Web hierarchy and collected their characteristics. This process resulted finally in a collection of 2500 to 10000 observations for each of the servers involved in the experiment.

The other aspects of characterizing the WWW application objects were on the client side. The experiment involves collecting packet traces using NetVCR from an user environment. The student and staff access subnetworks at the University of Karlskrona/Ronneby have been used for this experiment. The results reported here are based on collections done during the months of June and September 1999. The university user base is quite small and thus the observations generated per day is small. Over these time periods, a total of about 18000 ob-

servations were collected for the staff access subnet and 20000 observations were collected for the student subnet.

4 Capturing HTTP Flows

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. Towards this objective, the basic capabilities of the ***tcptrace*** utility [19] have been augmented to capture the message timings as well. The message times are in the form of the standard UNIX epoch based timing principle. The ***augmented tcptrace*** utility accepts as the input the dataset collected by NetVCR and produces two outputs. First, it gives a summary output for all connections present in the given NetVCR dataset. Secondly, it outputs the application level data exchanges (and the associated timings) into separate files. For each TCP connection, two files are produced, namely one for each half of the connection. The summary output contains identification information for all the connections. Each connection is represented as one line in the summary giving information such as start and end epochs, the data bytes and packet counts that were sent in both halves of the connections, and the filenames that contain the actual data exchanged and their associated epoch timings.

Armed with the ***augmented tcptrace*** utility to map the basic data stream into TCP flows, the information from a large number of HTTP flows has been collected (as mentioned above). Towards this end, a second tool has been developed to help summarize the huge information generated from the flows. This tool, which is named the ***http_flowstat*** utility, is used to collect information about the application level objects and the protocol exchanges between the client and the server. Details about the ***augmented tcptrace*** utility and the ***http_flowstat*** utility are presented in [11].

The HTTP protocol can use both persistent (HTTP/1.1) and non-persistent (HTTP/1.0) mode connections. The protocol operation in the non-persistent mode transfers one Web object per TCP connection. In addition, the TCP connections to fetch the objects belonging to a given Web session may operate in parallel. Therefore, it is extremely difficult to identify a Web session by monitoring flows on the network. However, it is possible in this case to look into the content portion and determine the content size and content type of the individual Web objects but still there is no way to distinguish a *main page* fetch from an *embedded page* fetch. An URI that appears as an *embedded item* inside some *main page* could also have been explicitly requested by a user by specifying the URI associated with the specific object. Also, if a fetch of an *embedded item* is identified, it is not possible to associate a particular fetch to a particular *main page* activity. This is due to the fact that there could be more that one user asking for the same *main* or *embedded page* concurrently. In the literature, there are some results reported about the structure of Web pages where the client side browser processes the logs and makes assumptions that *embedded item* fetches should appear within milliseconds of the *main page* fetch [3]. However, this issue is not as clearcut as it appears. This problem has been resolved in our experiments by actually probing several thousands Web pages for their structure and suitably modeling this aspect [21].

On the contrary, in the persistent mode, the server does not close the TCP connection after sending the response. The *main page* and all the *embedded items* are likely to travel on the same connection. In such a case, it is quite possible to clearly determine diverse specific pa-

rameters like the Web page structure, the session duration, etc. However, there are parameters on the server side (e.g., timeout, maximum number of transfers) that are configurable even during the persistence mode operation. For instance, if there is no activity on a connection for the timeout duration, then the server will take the pre-emptive action of closing the connection in spite of the persistence mode. There are two submodes of operation in the persistent case: ***persistence with pipelining*** and ***persistence without pipelining***. In the pipelined implementation, the client sends the requests for all the *embedded items* without waiting for any responses from the server side. This simplifies the procedure of analyzing the structure because the number of transactions within the session can simply be obtained by counting the number of requests seen on the flow. However, in a general mixed scenario of various types of browsers, diverse server software implementations, as well as various configuration parameters, the mechanism of extracting information about the transactions within a session may become very complicated. As in the case of non-persistence mode operation, this brings an element of uncertainty into the data collection process (from the NetVCR dumps). As a consequence, the direct probe method of determining the Web page structure has been adopted for our purposes.

5 Direct Probe Method

In addition to the regular browser based interface of accessing the WWW information, occasionally the Web can be also accessed via Web mirroring software (sometimes also known as **robots**). The primary purpose of this software is to duplicate the entire content tree of a designated Web server on the local disks thus completely mirroring the site. This method is often adopted to create proxy contents in order to improve the performance. Given an URI, such 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 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 also a selectable parameter.

For the purposes of analysis work, a mirroring software has been used. The software is available in the public domain and is known as **GetWeb** [25]. The intended use of this software is for downloading Web pages and restructuring them so that the downloaded pages can be browsed offline. The software has been also modified in the sense that it downloads the content pages and reports the content types and the sizes [11]. It also counts the number of embedded references 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 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 the properties are relevant in characterization of the WWW service.

6 HTTP Characteristics

Using the software tools described above, the statistics about the key elements have been collected. Some of the most important elements are as follows:

- Session arrivals. These statistics indicate the intensity of WWW requests generated by users.
- Number of transactions within a session. This is essentially the number of embedded items inside the Webpage that constitute the session. Almost all WWW sessions have multiple transactions. This is due to the way the content of Web pages are organized. This property further determines the session durations. The larger the number of transactions, the larger is the number of transfers that have to be performed to complete the session.
- Request message lengths. The request messages from the client to the server are usually short messages that are transmitted for fetching each of the Web objects whether they are of *mainpage* type or of *embedded* type. These messages and their timings determine the traffic stream that is generated from the client to the server.
- Reply message headers. The WWW reply messages from the server to the client consist of a header portion and the content portion. The statistics of the reply header sizes themselves is an important item that needs to be characterized for evaluating the traffic created in addition to the content portion. Furthermore, the reply headers themselves have important information that needs to be collected and characterized.

Some of the most important attributes that can be generated by studying the reply headers are as follows:

- The success or failure statistics of WWW access requests, which can be generated by using the reply code information in the reply message header.
- The percentage of the requested items that were not transferred due to caching. This can be obtained by studying the *Last-Modified* field in the reply header.
- The type of the object transferred. When the client side sends a request, the only information it has is the URI. The classification of the objects to HTML or non-HTML can be done only by studying the *Content-type* field in the reply header from the server.
- The content lengths. This is the main contributor to the traffic from the server to the client.

All these statistics have been collected (as described in section 3) and the analytical results are reported in section 8.

7 Modeling Methodology

Detection and estimation of heavy-tailed properties in the distributions of application layer objects is an important aspect in performance evaluation of applications. It may for instance reveal the presence of infinite mean or variance. Accurate estimation of the exponent is also important as it captures the degree of Long-Range Dependence (LRD) inherent in the objects [6]. Such estimates are also very useful in building simulation models that can reproduce traffic conditions as observed in real networks.

Often the random variable possessing heavy right tail appears hidden behind another distribution. While the two distributions may have very different tail behavior in a mathematical sense, it may be quite difficult to segregate the two in a practical fitting problem. For instance, it has been observed that the sizes of HTTP application objects have usually a distribution that can be described as a Lognormal one and this can contribute with about 80 to 95 percent of the probability mass with the remaining contributed by a Pareto distributed random variable. The crux of the problem lies therefore in determining the cutoff point between the two distributions [6].

The procedure followed for fitting probability distributions for mixture models is as follows [11]:

- First step is to look at the histogram and empirical distribution plot and the summary of the observations (such as minimum, maximum, mean, median, variance, etc). This step helps in eliminating many candidate distributions. This may also help in determining if the observation data set can be modeled using a single distribution or mixture models are required.

- Visual techniques such as Hill estimator plot [22] and the Complementary Cumulative Distribution Function (CCDF) plot on a log-log scale are used to inspect whether the set of observations has a heavy tail. The heavy tail shows up as a straight line in the CCDF plot. On the Hill plot, the initial instability quickly dies down and the curve becomes roughly parallel to the x-axis. In the case of mixture models, the parallel behavior is maintained for the entire tail portion. If the distribution of the body portion is Lognormal, then the curve will start rising just after the cutoff point. While it may not be possible to determine the cutoff point accurately, this does provide a hint about the tail mass.

- The next step is to try out some candidate distributions and perform the null hypothesis test to find out the significance level of the observations having been drawn from the candidate distribution. The Anderson-Darling (AD) test provides an adequate framework for this [7]. In the case of mixture model, a method of successive right-censoring is used to partition the observations. The range for the amount of right-censoring is decided by using the hint from step 2 above. The AD test needs also to be modified to be applicable to censored samples [7]. If the test does not show any significance level when applied to the complete set, the test is applied then to smaller subsamples. In each case 100 tests are performed and the number of times are counted when the 1% and 5% significance levels are passed. The null hypothesis is considered to be passed if there are at least 50 trials showing 5% significance and at least 75 trials showing 1% significance. If the sample does not meet this significance criteria, then the process is repeated with

smaller subsamples. For the HTTP samples, it was observed that the maximum subsample size that met the significance criteria was 1000 and the minimum was 100. In the case of mixture models, a suitable cutoff point was selected where the null hypothesis is satisfied best for both the participating distributions. This was done by using different amounts of censoring and evaluating the significance criteria. In some samples, the significance criteria was very difficult to get. In summary, it is true that the null hypothesis test was not passed cleanly and thus the model is not exact in a statistical sense. At the same time, having shown some limited significance levels, it can be stated that the model is close to the specific distribution mixture.

- The parameters for the distributions that passed the null hypothesis in step 3 above are obtained using the Maximum Likelihood Estimation (MLE) method. Special methods applicable to estimate parameters from censored samples are also used [13].

- Having obtained the distributions and the parameters thereof, a measure of the discrepancy between the observations and the model can be obtained using the λ^2 criteria [20].

8 Traffic Analysis

WWW is a server-centric application that 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 this experiment is 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 are collected using the ***http_flowstat*** program. The main characteristics of the WWW application are as follows:

- The statistics of WWW session arrivals are determined by the user behavior. Data collected from the real Internet access links shows the aggregate behavior of hundreds of clients. The analysis of one day's collected traces using ***http_flowstat*** shows the generation of thousands of connections. A WWW session starts with a *mainpage* request and may contain several *embedded item* requests as subsequent transactions. However, as explained above, it is extremely difficult to distinguish between *mainpage* and *embedded item* requests by only analyzing the flows captured from the network. However, in a few cases, the silence period between successive WWW requests are large enough to make this distinction. Such inter-arrival times were collected from the output of the ***http_flowstat*** program. Any inter-arrival gap larger than 5 seconds was assumed to be an inter-session gap. As per the observations so obtained, the arrivals of WWW sessions can be reasonably well modelled by a renewal process, and in most of cases they are simply a Poisson process [11]. Alternatively, the inter-arrival time between *mainpage*

requests is well modeled by an Exponential process with a mean of 15 seconds. This parameter influences in fact the inactive OFF time of the WWW client side.

- The inter-arrival time gaps between successive WWW requests belonging to a given WWW session, which represent the active OFF times of the WWW client (fig. 1). This parameter is obtained from the output of the ***http_flowstat*** software and it has been clearly observed that these timings are in the range 100-500 milliseconds. These timings are mainly dependent on the HTTP version and the pipelining feature of the browser software. With pipelining, the client software transmits all the requests one after another without waiting for the results of the previous request to arrive. In such cases, the active OFF times tend to be very small. This is the case with HTTP/1.1 and HTTP/1.0 with persistence mode operation. Even in case of HTTP/1.0 with no persistence, the client may open several TCP connections in parallel and send out one HTTP request on each of these connections.

- The request message sizes were collected from the output of the ***http_flowstat*** program as well. The analysis of the message sizes reveals that this parameter can be well modeled as an Uniformly distributed random variable in the range 180-600 [11]. Figure 2 shows the structure of the request headers. For a browser like NETSCAPE, the limited variability seen in the sizes of request messages is mainly contributed by the first line showing the URL, the second line containing the caching information, and the fifth line containing the domain name of the server. Similarly, the header sizes of the response messages also show limited variability. This parameter can be modeled by an Uniform random variable in the range 250-700.

- The results for the fitting process for different server categories are presented in table 1. The CCDF plots for the server categories are shown in figures 4 to 6. The Web pages have been fitted as a mixture of Lognormal (with parameters μ and σ) and Pareto distributions (with parameters α and β). The quality of fitting for the two candidate distributions are shown in the 'Remarks' column of table 1. The remarks (Lok) and $(Lpoor)$ refer to the Lognormal body portion. Similarly, the remarks (Pok) and $(Ppoor)$ refer to the Pareto tail. These remarks are based on the significance levels passed by the censored AD test. The choice of subsample sizes for applying the AD test in each case is also shown. The collected observations have been grouped into HTML and non-HTML types and their relative frequencies are also shown. It is observed that this model does not fit well in some cases. In a majority of cases however, it appears that the tail portion comes from a Pareto process. However, in some cases (two of the news/media servers), the power-law behavior is negligible. In such cases, the complete distribution can be well represented as a Lognormal random variable only. Such cases are highlighted with a '*' in the 'Remarks' column. Furthermore, in the case of commercial organizations, the Lognormal distribution seems to be a poor choice to fit the body portion. This is a disturbing result and as such this will be further investigated. The HTML and non-HTML groups, when plotted separately on the CCDF plot, reveal that this behavior appears to come from the HTML type documents mostly (figure 7). From the CCDF plot it also appears that in such cases the body portion can be approximated by an Uniform distribution. This feature seems to be common to all the commercial servers investigated.

- The analysis of the HTTP object sizes collected on the client side via NetVCR monitor is shown in table 2. This is a typical case seen in the access networks (staff and students) in the Unversity College of Karlskrona/Ronneby. A representative set of results are shown in the table. The CCDF plots for the sizes of Web pages accessed in the two subnets is shown in figure 8. A notable feature of the results shown here is that the qualitative aspects of the model fitting are much better than shown in table 1 and hence they are not been reported.

- A deeper analysis of the traffic seen on the staff access network is shown in figure 9. The density of HTTP traffic on this network is rather low. In addition, the access pattern is rather repeatative. In a broad sense, every day the traffic is drawn from a few servers and almost the same set of pages. A further investigation reveals that the major contribution to the traffic comes from a basic set of Web servers that includes newspapers, technology sites, etc. The caching percentage is about 10%. The success rate of Webpage access is about 95%. When the Web pages are grouped and studied separately (figure 9), then it is seen that the HTML type pages accessed on the staff network have negligible power law behavior. The non-HTML pages show some power law behavior but the intensity is quite low. This may be attributed to the low variability in the Web pages accessed on the staff network.

- The density of HTTP traffic on the student access network is expectedly higher (figure 10). On a typical day, the student net may show about 350 MB of HTTP traffic. In addition, the analysis of the access pattern shows a richness in variety both in number of Web sites and content types (such as audio, graphics, java). The caching percentage is lower in comparison to the staff access net and it is about 4%. The success rate of Webpage access is about 90%. When the Web pages are grouped and studied separately (figure 10), the higher intensity of power law behavior is apparent in both the HTML and non-HTML categories.

Site Name	μ	σ	α	β	Tail mass	Signif sub size	λ^2 value	cutoff point (byte)	HTML %	non HTML %	Remarks
www.cs.berkeley.edu	8.49	1.86	0.766	5717	0.13	200	0.06	81435	44	56	$(Lok)(Pok)$
www.hk-r.se	7.97	1.27	0.624	1160	0.24	200	0.09	11394	58	42	$(Lok)(Ppo)$
www.cs.purdue.edu	7.09	1.27	0.83	1547	0.15	200	0.06	15137	74	26	$(Lpo)(Pok)$
www.cam.ac.uk	8.12	1.28	1.03	1860	0.16	200	0.02	10964	68	32	$(Lok)(Pok)$
www.uiuc.edu	8.01	1.30	1.62	3865	0.17	200	0.04	11526	69	31	$(Lok)(Pok)$
www.hp.com	-	-	0.46	54	0.04	100	0.12	53658	40	60	$(Lpo)(Pok)$
www.ibm.com	-	-	0.63	418	0.06	100	0.13	36096	61	39	$(Lpo)(Pok)$
www.nokia.com	-	-	0.51	142	0.06	100	0.13	35428	34	66	$(Lpo)(Pok)$
www.lucent.com	-	-	1.12	16578	0.06	100	0.14	199961	60	40	$(Lpo)(Pok)$
www.rediff.com	8.50	1.56	-	-	-	100	0.05	-	67	33	*
www.nytimes.com	8.85	1.88	1.85	8423	0.29	100	0.17	16408	72	28	$(Lpo)(Pok)$
www.the-week.com	9.78	0.64	-	-	-	100	0.05	-	34	66	*
www.india-today.com	8.90	1.53	1.46	6160	0.13	300	0.05	24824	50	50	$(Lok)(Pok)$

Table 1: Modeling results for HTTP object sizes collected from different categories of Web servers

9 Structure of Web Pages

The structure of the content pages is likely to be a property of the server itself. This feature represents the qualitative design aspects of the content itself and to that extent reflects the

Karlskrona	Type	μ	σ	α	β	Tail mass	Signif sub size	λ^2 value	cutoff point	%
Staff Net	HTML	8.54	1.60	-	-	-	300	0.06	-	45
	non HTML	7.71	1.68	1.69	3178	0.14	300	0.04	10112	55
Student Net	HTML	6.64	1.50	1.30	1418	0.14	200	0.04	11394	30
	non HTML	7.08	1.87	0.93	1147	0.22	200	0.06	5802	70

Table 2: Modeling results for the HTTP object sizes collected through packet traces

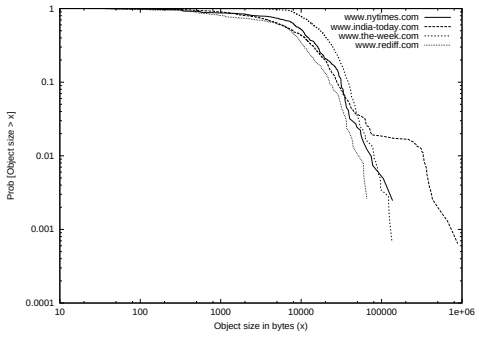

Figure 4: CCDF of entertainment servers

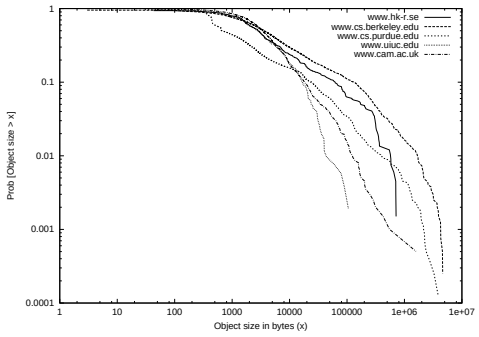

Figure 5: CCDF of educational servers

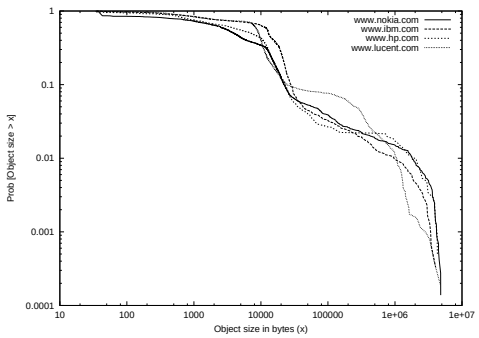

Figure 6: CCDF of commercial servers

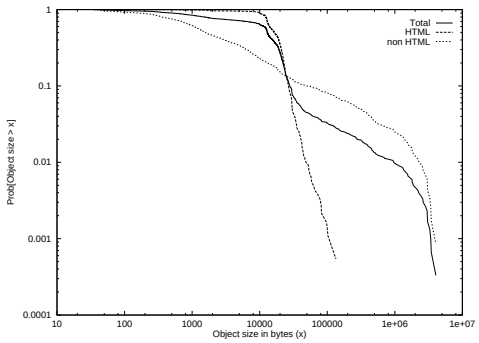

Figure 7: CCDF of ibm.com server

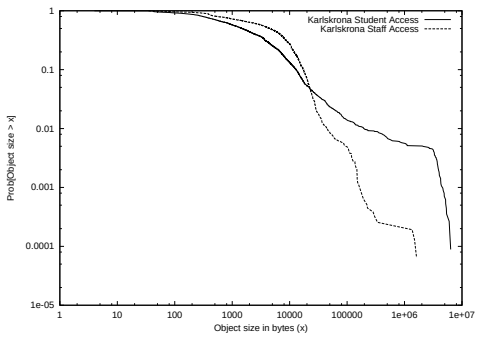

Figure 8: CCDF of Karlskrona/Ronneby

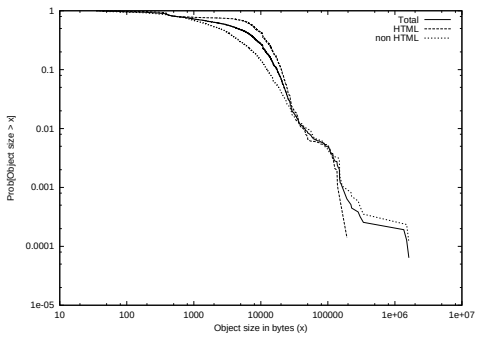

Figure 9: CCDF of Karlskrona/Ronneby, staff only

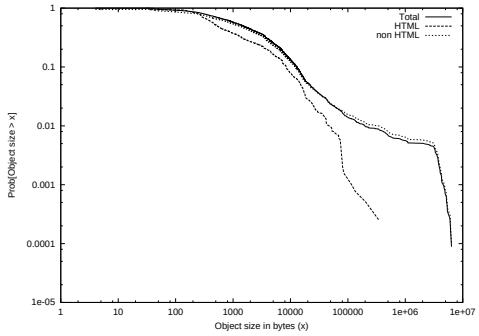

Figure 10: CCDF of Karlskrona/Ronneby, students only

properties of the specific organization. Intuitively, it is expected that the Web page design features at the universities may be different from those designed for commercial purposes. It is expected that the differences are both quantitative and qualitative. For instance, the density of *embedded items* is expected to be higher in the case of news/entertainment and commercial sites. On the other hand, the bulk of pages on newspaper sites will be of HTML type and in terms of sizes they should be small. As far as the university sites are concerned, these are expected to be rich with research reports having Postscript or PDF format. These files tend to be large in size and therefore contribute towards the power-law behavior in the tail. While these intuitive arguments are difficult to prove rigorously (because of the subjective human factors involved), they are, to a limited scale, demonstrated in figures 4 to 6.

Based on these intuitive principles, attempts have been done to build up a stochastic model for the number of *embedded items* that may occur in a Web page. It is recalled that each of the *embedded items* is a transaction within the WWW session. The following steps describe the modeling methodology for this important element in WWW traffic characterization:

- The basic dataset used in the modeling process is the output of the ***GetWeb*** software that probed a large number of WWW servers and recursively downloaded their content pages.
- The data generated in the previous step was grouped using two levels of indexing. First level of index is the *serverid* and the second level is the *pageid*. The design of the indexing mechanism was geared towards the goal of retaining the server information and also the particular page information within the specific server. It was observed that in many cases a single *embedded item* occured many times (e.g., the GIF files representing the bullets in a HTML page layout). The modeling process counts the contribution of such multiple occurances as 1. In other words, the modeling procedure only considers unique *embedded items* and rejects the duplicate occurances of the *embedded items*. This corresponds to a realistic scenario because after the GIF object is downloaded the first time, it is available in the local cache, and hence it is not downloaded again.
- Subsequently, the literature in applied statistics was searched to find the statistical modeling of a real-life situation that closely resembles the problem at hand. For instance, a target situation was found to be the modeling of the numbers of defects per square yard

in fabrics [24]. Conceptually, a piece of fabric may correspond to the WWW *mainpage*, whereas the defects on the specific piece of cloth may represent the *embedded items*.

Based on these facts, a mathematical model for this phenomena has been built [15]. The number of *embedded items* has been found to be well modelled by a Negative Binomial distribution (Fig. 11). This distribution is used, for instance, to find the number of Bernoulli trials needed to find the r-th success. Furthermore, it has been also shown that the Negative Binomial distribution can be well approximated as a mixture of Poisson distributions such as the expected values of the Poisson distributions vary according to a Gamma distribution [12].

A simple model of the number of embedded documents in a Web page is as follows. Assume that, on a specific server i, the number of *embedded items* in the Web page j is $X_{i,j}$. This number is conditioned by the intensity λ_i, which is a server characteristic that represents the density of presence of *embedded items* in the Web pages resident at server i. In other words, the quantity λ_i is implicitly fixed when the user selects a particular server during a Web session. Accordingly, the (conditional) distribution $F(X|\lambda)$ of the number X of *embedded items* in a Web page can be modelled as:

$$F(X|\lambda) \sim P(\lambda) \tag{1}$$

where P represents the Poisson process. From the collected observations (regarding the total number of *embedded items* occuring in a page), the number of pages having zero *embedded items* is excluded. Consider C as being the fraction of these pages. The remaining observations (having the number of *embedded items* greater than zero) can be further used to generate a distributional model by using Gamma distribution with shape parameters α and β. In other words, the random process X occurs as a mixture of Poisson distributions and the expected intensity λ of every Poisson distribution vary according to a Gamma distribution with PDF:

$$f(\lambda) = \frac{\lambda^{(\alpha-1)} exp\left(-\frac{\lambda}{\beta}\right)}{\beta^{\alpha} * \Gamma(\alpha)} \tag{2}$$

where $\Gamma(\alpha)$ is the Gamma function (also known as Euler function) of variable α and the parameters λ, α and β are larger than zero. Under these conditions X has a Negative Binomial distribution (with parameters α and β) and the PDF of the number of *embedded items* is [12]:

$$Prob[X = x] = \int_0^{\infty} f(\lambda) \frac{\lambda^x exp(-\lambda)}{x!} d\lambda = \binom{\alpha + x - 1}{\alpha - 1} \left(\frac{\beta}{\beta+1}\right)^x \left(\frac{1}{\beta+1}\right)^{\alpha} \tag{3}$$

A model for the generation of X is as follows:

- Generate an Uniformly distributed random variable ω in the range 0-1. The value C can be estimated from the observations (direct probe) and in our case it has been estimated to be 0.1202. If $\omega \leq C$, then return the number of *embedded items* as zero. Otherwise continue with the next step.

- Generate a random number N from the Negative Binomial distribution. From the collected data it was observed that the maximum number of *embedded items* in a page is 75. Also, there were about 6% of the cases where the *embedded items* were located outside the server hosting the *mainpage*. Using these observations, the value of the parameters of the distribution obtained via the maximum likelihood estimation are $\alpha = 1.386$ and $\beta = 6.985$.

Fig. 11 shows the observed CDF of the number of *embedded items* (as collected from real Web pages) against the distribution function obtained by the modeling process presented above.

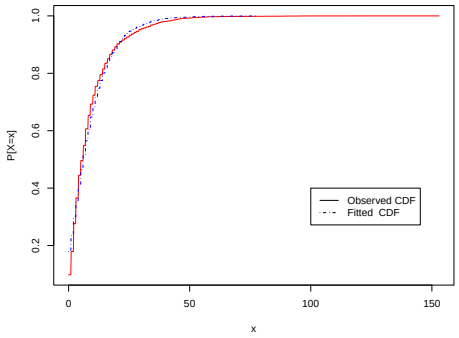

Figure 11: CDF of the number of embedded items

10 Summary

A study for the characterization of WWW clients and servers has been presented. Detailed results have been reported on measuring, modeling and analysis of HTTP traffic collected from different classes of sites. The analysis of the collected statistics reveals two distinct sets of properties, namely the latent properties and the exhibited properties. The latent propeties refer to the inherent property of the various types of Web servers. Similar statistical properties are evident among the servers belonging to the same class. The media related group, the commercial companies group, and the universities group show similar distributional properties in their object sizes. On the contrary, the exhibited properties include the sizes of the actual objects that are downloaded by the users, the request and response headers, the caching phenomena, the success or failure probabilities, etc. Object sizes for both the latent and the exhibited cases were separately modeled. It was observed that the exhibited properties may be different from the latent properties. Last but not the least, a distributional model has been presented to accurately model the number of *embedded items* that occur in WWW pages.

Acknowledgment

We would like to thank Dr. T. V. Kurien of Niksun Inc. for numerous useful discussions on the topics of statistics and modeling.

References

[1] Abdullah G., *Analysis and Modeling of World Wide Web Traffic*, PhD thesis, Department of Computer Science, Virginia Polytechnic Institute and State University, May 1998.

[2] Arlitt M. and Williamson C., *Internet Web Servers: Workload Characterization and Performance Implications*, IEEE/ACM Transactions on Networking, Vol 5 No 5, October 1997.

[3] Barford P. and Crovella M.E, *Generating Representative Web Workloads for Network and Server Performance Evaluation*, Report BU-CS-97-006, Computer Science Department, Boston Univerdity, 1997.

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

[5] Crovella M.E. and Bestavros A., *Explaining World Wide Web Traffic Self-Similarity*, Technical Report: TR-95-015, Computer Science Department, Boston University.

[6] Crovella M.E. and Taqqu M.S., *Estimating the Heavy Tail Index from Scaling Properties*, Methodology and Computing in Applied Probability, Vol 1, No. 1, 1999.

[7] D'Agostino R.B. and Stephens M.A., *Goodness-of-Fit Techniques*, Marcel Dekker Inc., 1986.

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

[9] Gettys J. *HTTP problem statement*, http://www.w3.org/Protocols/HTTP-NG/951005Problem.html, 1995. W3C.

[10] Heidemann K., Obraczka K., and Touch J., *Modeling the Performance of HTTP Over Several Transport*, IEEE/ACM Transactions on Networking, Vol 5.5, October 1997.

[11] Jena A.K., *Modeling and Analysis of Internet Applications*, Licentiate Thesis, University of Karlskrona/Ronneby, Sweden, 2000.

[12] Johnson N., Kotz S., and Kemp A.W., *Univariate Discrete Distributions*, Second Edition, John Wiley & Sons, 1993.

[13] Johnson N., Kotz S., and Balakrishnan N., *Continuous Univariate Distributions*, Vol. 1, Second Edition, John Wiley & Sons, 1994.

[14] Kristol D. and Montulli L., *HTTP State Management Mechanism*, RFC 2109, February 1997.

[15] Kurien T.V., correspondence with T. V. Kurien (Niksun Inc.), 1998.

[16] Kurose J.F. and Ross K.W.M., *Computer Networking A Top-Down Approach Featuring the Internet,* http://www.seas.upenn.edu/~ross/book/Contents.htm, October 1999.

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

[18] Nielsen et al, *Network Performance Effects of HTTP/1.1,* SIGCOMM 1997.

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

[20] Paxson V., *Empirically-Derived Analytic Models of Wide-Area TCP Connections*, IEEE/ACM Transactions on Networking, Vol. 2, No. 4, August 1994.

[21] Pruthi P., Jena A.K. and Popescu A., *HTTP Interactions with TCP*, Proceedings of the 11th ITC Specialist Seminar: Multimedia and Nomadic Communications, Yokohama, Japan, October 1998.

[22] Resnick S.I., *Heavy Tail Modeling and Teletraffic Data,* The Annals of Statistics, 25(5), 1997.

[23] Ross K.W., *Hash-Routing for Collections of Shared Web Caches,* IEEE Network Magazine, November 1997.

[24] Scheaffer R.L. and McClave J.T., *Statistics for Engineers,* PWS Publishers, 1982.

[25] Xanthakis S., *GetWeb version 2.7.2,* January 1999, http://www.enfin.com/getweb/