Design and Use of an aggregated HTTP Traffic Model*

Kleber V. Cardoso and José F. de Rezende
Grupo de Teleinformática e Automação
COPPE/Programa de Engenharia Elétrica
Universidade Federal do Rio de Janeiro
{kleber,rezende}@gta.ufrj.br

Abstract

This paper presents a new HTTP traffic model based on the aggregation concept. The model development, evaluation and application are shown. In addition to the basic HTTP traffic characteristics, the traffic model has an easy and accurate load control. Some examples are provided to present the traffic model usage.

Keywords: HTTP Traffic, Internet, Quality of Service (QoS), Performance Evaluation, Self-similarity.

1 Introduction

In the last years, web has maintained its status of the Internet killer application and there is not clues that the situation will change soon. The HTTP, responsible to transfer web content, dominates the traffic traces. According to recent statistics from CAIDA [1], HTTP typically represents from 47% to 69% of the bytes sent over the Internet. The number of services and the amount of information available in the web keeps growing and this looks like to be a dominant trend for some years. First, because web is a suitable application for any kind of service which is based on text and graphics. Second, HTTP is adequate to transfer different types of files, from small Java applets to huge non-stream videos. Third, and most important, web has become a kind of universal interface. The simple and friendly "look and feel" of the web pages have allowed different services and information to be widely available to almost any system regardless the hardware or the operating system.

Within this context, it is important to understand how HTTP traffic behaviors in order to understand and make improvements in the Internet. One way to do this is developing and using HTTP traffic models. Many works have been made in this area [2, 3, 4, 5, 6], using different approaches in the development. The majority proposes models that describe a common web client behavior [2, 5, 6, 7], with different levels of details. A small number of works [3, 4] focuses on the behavior of a group or aggregation of web clients, which has as main advantage the simplicity. These works do not present precise methods to control the network load generated by their models. In many cases this is a wanted characteristic since it can represent a control over network condition. In many situations, the lack of examples precludes people to utilize the existing models, which drives to repeated job on traffic model development. This paper proposes improvements in these subjects.

This paper proposes a new HTTP model, which is based on the concept of aggregated behavior and presents two main advantages: small number of parameters and easy and precise load control. The development and features of the model are detailed. The procedures for

---

evaluation of the model properties are also shown. Examples of the model utilization are shown and results are presented and discussed.

This paper is organized as follows. Section 2 reviews the HTTP protocol, traffic collecting and modeling. Section 3 presents a new HTTP aggregation model and its features. Section 4 shows some uses of the generator based on the proposed traffic model. At last, in the section 5, conclusions and final comments are drawn.

2   Background

This section presents HTTP information that is important for traffic modeling. It is introduced the two main approaches of traffic modeling: client and aggregation. The benefits and short-comings of each one are shown. The techniques widely used on traffic modeling and collecting are also presented.

2.1   HTTP Protocol

HTTP is a request-reply protocol designed to transfer web files. Each transfer consists of file requests for pages or objects from a client to a server and corresponding reply or an error notification. In a simpler and general way, the user requests to the browser a URL (Universal Resource Locator), or just a site, and the page is requested from the browser to the server. When receiving the page, the browser parses it to find the component objects, such as images, sounds, Java classes, etc., and then it requests these objects to the server. The user has now available the information and links to other pages. If the user selects any link the process starts again.

The first HTTP version was HTTP/0.9 and consisted of a simple protocol for raw data transfer across the Internet. HTTP/1.0, as defined by [8], improved the protocol by allowing messages to be in the format of MIME-like messages, containing meta-information about the data transferred and modifiers on the request/reply semantics. MIME enables browsers to display or output files that are not in HTML format, such as graphic, audio and video. However, HTTP/1.0 does not sufficiently take into consideration the effects of hierarchical proxies, caching, the need for persistent connections, or virtual hosts. In addition, the proliferation of incompletely-implemented applications calling themselves "HTTP/1.0" has urged a proto-col version change in order to two communicating applications to determine each other's true capabilities. HTTP/1.1 was designed to fulfill these gaps [9].

HTTP communication usually takes place over TCP/IP connections. The default port is TCP 80, but other ports can be used. This does not prevent HTTP from being implemented on top of any other protocol on the Internet, or on other protocol architectures. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used. In this paper, only TCP will be treated since it is the reliable transport protocol of common use in the Internet. Usually, the HTTP protocol features presented in this section are emphasized in traffic modeling.

2.2   HTTP Traffic Modeling

Simulation is a widely used tool for computer networks evaluation, but it is important to have suitable traffic models to get useful results. The majority of works about web traffic modeling has concentrated on developing client models [2, 5, 6, 7], which focus on the behavior of individual web clients. Other approach is to model the aggregation behavior of several web clients, i.e., an aggregation model.

Both models have advantages and shortcomings. The client model is able to capture more details of the application, so it is in some sense a better mimic. However, this higher level of detail

brings more complexity to the model because it demands the understanding and configuration of more parameters. In some situations the level of detail does not help in the evaluation, since many of the details simply does not matter. Some examples of that sort of evaluation are the ones performed in bottleneck links, such as resource provisioning, queue behavior, some scheduling mechanisms, etc. In these cases, simpler models are a better choice.

The aggregation model is generally a coarser approximation of the real traffic. In spite of this, its simplicity allows it to simulate some conditions and identifying behaviors that are difficult with client models. In addition, client models tend to consume more computing resources than aggregation models when representing a large number of web clients in a simulation environment. In both kinds of model an important issue is the choice of application's characteristics that are desired, since they are the focus of the model development. Some examples of these characteristics are burstiness, network load, long-range dependency, etc.

A model (aggregation or client) utilizes parameters to reproduce certain properties of the web application. Some examples of parameters are transfer size, interval between pages, number of objects per page, etc. To describe these parameters two approaches are used: one based on real traffic samples and other analytic. The models created using these approaches are known as structural models [10], since they try to characterize the traffic nature.

The use of real traffic samples consists of describing a certain application parameter through a set of predefined values which are collected from a real network environment. The main advantage of this method is the easy of implementation and accurate representation of a known system. However, this approach treats the generated traffic as a "black-box". In addition, the generator traffic based on this kind of model becomes hard to set up since new conditions or variable demands are not easy to configure.

The analytic approach lies in the use of probability distributions to describe a certain parameter. A probability distribution tells how a sequence of random values behaves, provided that there is available enough number of samples. When the distribution is known, it allows to generate new and different sequences of values following such distribution. The main drawback of this approach is the difficulty found in identifying and configuring the distribution that describes adequately the sequences of random values of the application parameters.

A third approach can be included, which consists of using known abstract processes to try to capture only the statistical traffic properties with independence of the subjacent mechanisms of traffic generation. This approach is efficient and quite simple to implement. Moreover, this approach is useful when specific features are of interest. For example, self-similarity can be easy reproduced by a fBm process (fractal Brownian motion). However, this sort of method does not take into account important factors from the traffic profile and neglects elements such as the congestion control of TCP, which is an important feature of HTTP traffic. Models based on this approach are known as "behaviourist" [10].

2.3   Collecting Traffic

The development of traffic models for interactive applications normally demands a study of the effective traffic generated by these applications. In other words, to develop an HTTP traffic model is important to collect traffic samples from real network environments and extract the necessary information. The extraction of certain information from web traffic is not trivial and requires the development of sophisticated heuristics. It follows the main methods for collecting HTTP traffic and the information of most interest to develop aggregation and client models.

The **server logs** method is based on the native log feature of the web servers. The main benefit of this method is exactly this easy availability of the information, since log is a very common web server characteristic. On the other hand, in this method is not easy to capture the access patterns among multiple web servers. Another disadvantage is the lack of information about HTTP headers, which can represent a significant protocol overhead.

A method known as **client logs** consists of collecting information from instrumented browsers. This method was used in the past [11, 12], but nowadays it is difficult to implement because it requires the distribution of modified browsers for a quite large number of typical web users. Client logs method presents a hard way of identifying the aggregation behavior.

Packet traces is the method most used in recent works. In this method, packets or part of them are collected from a subnet transporting HTTP traffic, typically an Ethernet LAN or other shared media. This method does not present an easy way to identify some parameters such as the effective transferred number of objects per page and the interval between pages. However, several techniques (or heuristics) were created and some public domain tools were developed. HTML-REDUCE [3] and BLT [13] are examples of this kind of tool which have presented good results and have helped in traffic analysis and the development of models.

At last, the **proxies logs** method collects information in a manner similar to the last one. A minor disadvantage of this method is the proxy limitation, i.e. a proxy normally serves a unique LAN or a small group of LANs, while the last method is able to collect information from a transit network among several other networks. So the clients and servers heterogeneity of the last method tend to be higher.

After gathering the traffic it is necessary to identify and account the parameters of interest. They vary from model to model. Some parameters appear in several models such as page/object size, while other are strictly related to a certain model, for example the request size. Some parameters are easily measured and some require sophisticated techniques. In the following there is a brief description of the most common parameters and some typical values found while measuring real traffic.

3   The Traffic Model

Following the proposal presented in the last section, a good start point for model development is to establish its objective, i.e. what application it will describe and what the focus or features are intended for the model. So it is important to establish a profile before beginning the model development. Despite the simplicity, this methodology can minimize the development time, since it has a clear focus and try to avoid unnecessary complexities.

In this work, it was established that the model should have few parameters. In addition, the traffic generator based on the model would be used as input to bottleneck links. Thus, the model could ignore several details related to individual web clients since the appropriated aggregation behavior was kept.

Traffic generators, sometimes called workload generator, generally do not have a simple way to adjust the load. It is common to use the mean load generated by a client or a set of them to a specific network configuration. To vary the load, the number of clients are varied. However, if the network configuration is changed then it is necessary to recompute the new load. Moreover, in this conventional way, the mean load is measured during all simulation time and measurement in short intervals can be always far from the mean. Thus the objectives of the model are an easy way to adjust load and samples near to the mean in time intervals shorter than the whole simulation time.

According to queue theory [14], the concept of load or utilization factor can be written as

$$\rho = R/C$$

in which
R - (work) arrival rate, and
C - maximum rate or system capacity.

The work that a new customer [1] brings to the system is equal to service time it requires. So

---

[1]To avoid confusion with the word **client** that is used to refer to web software

if system has a unique server (e.g. a bottleneck link router) the load can be rewritten as
$\rho = \lambda \overline{x}$
where
λ - mean arrival rate of customers, and
$\overline{x}$ - mean service time.
Considering the context of HTTP traffic and bottleneck link, the last equation can be modified
to $\overline{x} = \overline{L}/C$
in which
$\overline{L}$ - mean transfer size, and
C - maximum rate or system capacity or link capacity.
Thus,

$$\rho = \lambda \overline{L}/C \qquad (1)$$

ρ is main adjustment parameter and is used to choose different load conditions. ρ describes
the time percentage that the system is busy given a measurement window. C is fixed to a certain
network configuration. $\overline{L}$ controls the mean size of web transfer. $\overline{L}$ may describe the size of a
web page/object if the interest is HTTP/1.0 without keep-alive or the size of group of pages
and objects if HTTP/1.0 with keep-alive or HTTP/1.1 are the protocols of interest. At last,
λ describes the connection arrival rate, which varies according to $\overline{L}$ in order to accomplish the
established ρ. Thus the arrival rate can be written as
$\lambda = \rho C / \overline{L}$
Since
$T = 1/\lambda$
describes the interval between connection arrivals, then

$$T = \overline{L}/\rho C \qquad (2)$$

The distributions that describe the parameter $\overline{L}$ have been widely studied [3, 15, 6, 16] and
there is some convergence. The majority agrees on a heavy-tail distribution to describe this
parameter and examples of configuration are in table 1. The label information is used for future
citations of the distributions.

Table 1: Some distributions that describe the parameter $\overline{L}$.

Distribution	Configuration	Label	Reference
Pareto	mean - 4100 shape - 1.95	HTTP-1	[16]
Pareto	mean - 4100 shape - 1.35	HTTP-2	[16]
Lognormal	mean - 4827 std. dev. - 41008	HTTP-3	[3]
hybrid: Pareto - 7%, Lognormal - 93%	mean - 1463000 shape - 1.1 mean - 27600 std. dev. - 59714	HTTP-4	[4]
hybrid: Pareto - 12%, Lognormal - 88%	mean - 10558 shape - 1.383 mean - 7247 std. dev. - 28765	HTTP-5	[17]

3.1 Study of Network Load

In this paper, the system will be always a router, but many concepts can be extended to other network equipments such as switches. Based on this, it is important to define what network load means. Sometimes, the network load refers to a bandwidth use, which can vary from 0 to 100% of all output link throughput. A more accurate approach is to consider the load as the time use of the router. To measure the network load as the time use of the system (router) is necessary to establish a measurement interval or window. This interval is a time quantity in which the network load is measured. Initially, the measurement interval can be arbitrary and vary from milliseconds until the whole simulation time. However, as it has been said, many times is useful to have time interval smaller than the whole simulation.

Based on these concepts and on equation 1, ρ represents the effective network load by occupying the system during a percentage of time of a certain measurement interval. E.g., if $\rho = 0,9$ (90%) and the measurement interval is 10 seconds, the system should be in use during 90% of this time, i.e., 9 seconds. Two main issues arise about the model.

First, it was established that in measurement intervals smaller than the whole simulation time, the load should get close to the mean value. This would be affected by transfer sizes because short-term transfers could fulfill smaller measurement intervals, while long-term transfers can present large intervals. In addition, HTTP uses TCP as transport protocol, which causes the transfers to happen in variable rates. This would create "distortions" on the sequences of system use. This way, the measurement interval would be affected again.

Second, the protocol TCP is reliable and retransmit lost packets mainly due to buffer overflows. If more than a copy of a packet pass by the network point where the load is being measured, load would be higher than ρ. By another hand, since the model is designed to bottleneck links, measurements are took place at this point, an thus the losses should occur at arriving in the buffer and duplicates would not account.

To evaluate the previous issues, simulations were done to verify the relation between ρ and the measured mean load in different intervals. The methodology applied in this work was based on [18]. In the simulations were used traffic sources with transfer sizes of 1, 5, 50, 500 and 5000 KBytes, and also sizes which obey the distributions shown in table 1.

The topology chosen to make the simulations is a kind of dumb-bell and is presented in figure 1. The buffer size of the bottleneck link follows the suggested by [19], which is $2 * Bw * RTT$, where $Bw = C$ and RTT is the longest round trip time in the network. TCP Reno was the implementation choice, since it is still one of most used. The queue discipline is FIFO and queue management is Drop Tail, i.e. the traditional configuration of a router. Packets come from s_1 and s_2 belong to two different traffic classes, that is, they are marked differently. In this section this is not take into account since there is no packet differentiation. This configuration is used to keep an uniform environment in all experiments, including the ones that have packet differentiation.

Figures 2(a) and 2(b) present the results when ρ is based on a bottleneck link of C and the link really has this capacity. Figures 3(a) and 3(b) show results based on the same link capacity as the last one, but the bottleneck actually has $10C$. These simulations are important because help on verifying the relationship between ρ and load. In figures 2(a), 2(b), 3(a) and 3(b) the load a mean measured after the first 50 seconds until the end of the simulation, which happens in 500 seconds. The beginning of the simulation is discarded in order to eliminate the transient.

Figures 2(a) and 3(a) show that transfer sizes of 1, 5, 50 and 500 KBytes present a good match for ρ and load. By another hand, transfer sizes of 5000 KBytes exhibit a difference between ρ and load from 70% when the link capacity is C. Experiments have shown that the reason is bust behavior of TCP combined with his rate control. Since this transfer size is large enough to maintain the system in use for a long time, the TCP has the opportunity to rise the transmission window beyond the bottleneck link capacity which makes buffer overflows. Losses
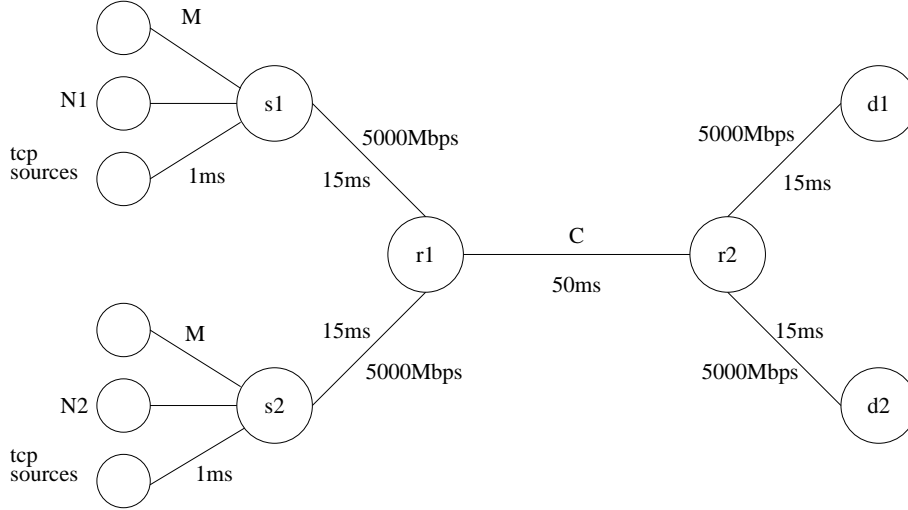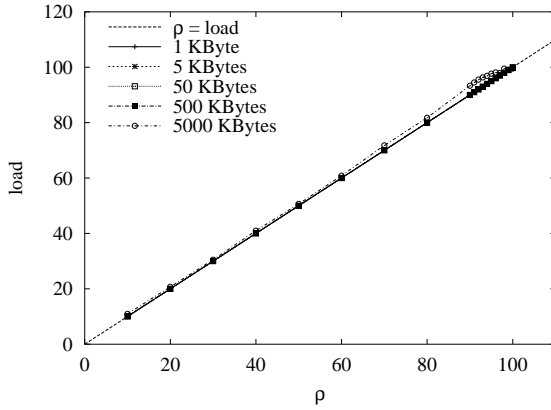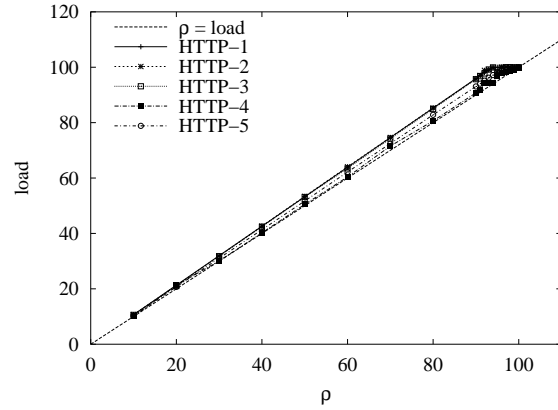
Figure 1: Topology.

(a) Fixed size transfers.

(b) HTTP transfers.

Figure 2: Load and ρ relationship in a bottleneck link of capacity of C.

demand retransmissions and decrease the goodput. This makes transfers to be stretched along the time. Thus, system gets a "debt" of idle time which it can not vanish until the simulation end because it happens abruptly when the simulation time reaches 500 seconds.

Table 2 helps understanding the phenomenon described. This table gives additional information about the traffic behavior when ρ is varied from 70% to 90%. $\rho = 70\%$ was chosen, since it is the start point of the "distortion", according to figure 2(a). In addition, were used two buffer configurations, one equal to figure 2(a) and other 10 times bigger. The last configuration intend to offer enough buffer space to accept long bursts. The table exhibits a significant higher number of losses when the buffer is B. It can be also noted a trend to increase in the number of simultaneous transfers as ρ rises. The time without active transfers is longer with buffer $10B$ than with B, which makes mean load not match ρ. Simulations have shown that $\rho = load$ when buffer is $10B$ as is the case when bottleneck is $10C$. The results are not presented here due to size limitations.

Since large transfer sizes can disturb the relation between ρ and load, it is important to
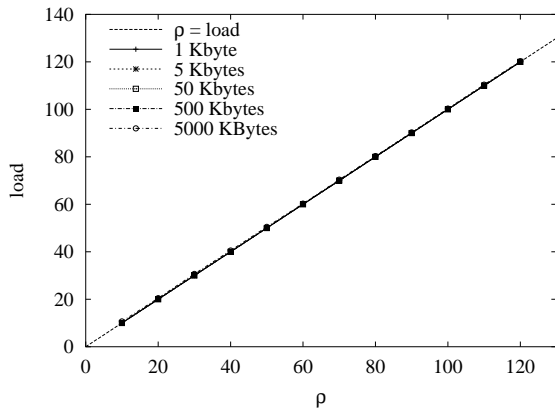
(a) Fixed size transfers.
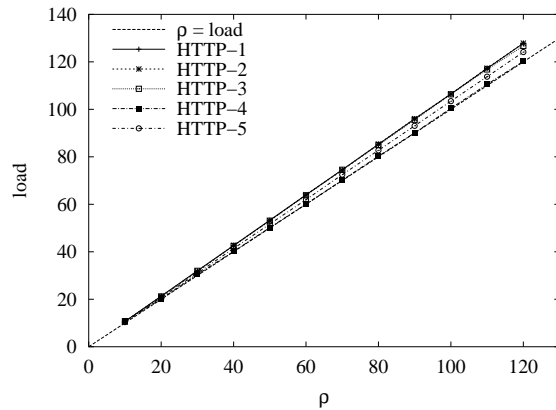
(b) HTTP transfers.

Figure 3: Load and ρ relationship in a bottleneck link of capacity of 10C.

Table 2: Details of 5MB transfer size with different buffer values.

	Losses (Packets)		No transfers (Seconds)		Simultaneous transfers	
ρ	B	10B	B	10B	B	10B
70	29679	0	36.01	100.01	2	2
75	21353	0	29.25	70.49	2	2
80	22854	0	0	51.30	5	2
85	21790	0	0	23.41	5	2
90	22256	0	0	1.31	6	2

evaluate in which rate this values appear in HTTP traffic. The evaluation was based on the transfer sizes distribution used in previous works widely cited. Table 3 exhibits the percentile of some typical transfer sizes. In this table is described the results of 100 sequences, with 100 thousand sample values each one. The distribution were based on [16], [4], [17] and [3]. As it can be seen, the long-term transfers happen rarely. It was also observed that these long-term transfers take place in a sparse manner.

Table 3: Percentile of some distributions.

	$< 5K$	$< 50K$	$< 500K$	$< 5M$
HTTP-1	83.30%	99.81%	99.99%	100.00%
HTTP-2	87.63%	99.45%	99.97%	99.99%
HTTP-3	85.38%	98.48%	99.94%	99.99%
HTTP-4	26.20%	86.63%	99.78%	99.96%
HTTP-5	73.18%	97.66%	99.96%	99.99%

Figures 2(b) and 3(b) show always $\rho < load$. Actually, the difference is small but for the sake of accuracy a detailed analysis was made. Surprisingly, the reason is only the NS simulator. In NS, the creation of each traffic source demands the configuration of the packet size. This packet

size is fixed and does not change whatever the size of data to be sent. So, if the transfer size is a multiple of the packet size then there is a good match between ρ and load. By another hand, there is always a packet which carries less data than it can and a padding is used to complete the size. This puts more bits in the network than is was previously established by ρ. E.g., if the packet size is 1000 Bytes and the transfer size is 1200 Bytes, then 2 packets of 1000 Bytes will be transmitted. Figure 4 shows the relation between ρ and load as transfer size varies from 500 to 100 Bytes and packet size is fixed in 500 Bytes. The results confirm what was expected, i.e. as the waste increases, load has the same behavior.
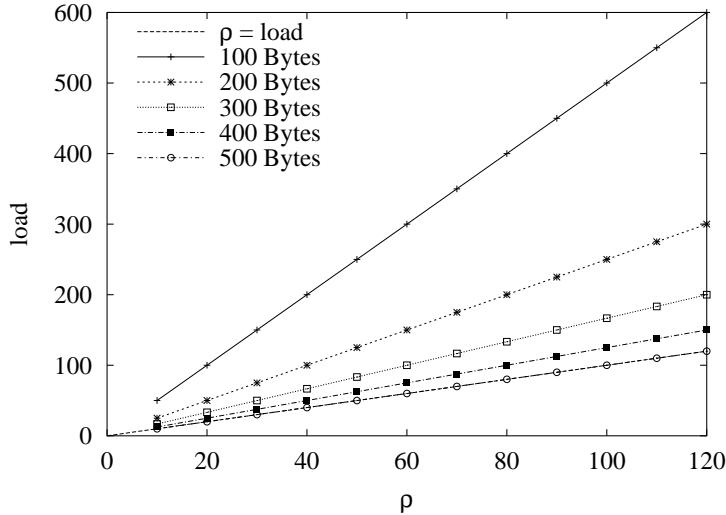
Figure 4: Load following the waste.

Another important part of the model is the measurement interval, since it was established as a model objective. To analyze how load varies in different measurement intervals, the mean was measured from 10 milliseconds until 50 seconds. These values are representative because they describe the ability of the model in controlling the load under different time intervals. Figures 5(a) and 5(b) show the mean load behavior when $\rho = 50\%$ and measurement intervals of 10 milliseconds and 1s. In these measurement intervals, the mean load presents significant variation. Figures 5(c) and 5(d) exhibit the mean load for different values of ρ and intervals of 10s and 50s. Under these intervals, the mean load presents a close match to ρ.

3.2 Simultaneous Connections

The model presents some interesting characteristics related to number of simultaneous transfers or connections. Initially, the model was based on HTTP/1.0, but by choosing suitable values for distribution of transfer sizes, HTTP/1.1 can be also resembled. Thus, the following evaluation will consider each HTTP transfer as a TCP connection, even though it can be modeling more than a page/object per connection. The use intended to the model does not care about this simplification.

TCP protocol gives a special contribution in the way number of simultaneous connections varies. First, thanks to slow-start algorithm, sequences of short-term transfers tend to have a high level of overlapping. Second, slow-start and congestion-avoidance algorithms help to increase the number of simultaneous connections as ρ rises. Figures 6(a), 6(b), 7(a) and 7(b) illustrate these situations. Figure 8 summarizes results from fixed size and HTTP transfers. It can be seen that as ρ gets closer to 100% the number of simultaneous transfers increase significantly, in special for HTTP transfers.
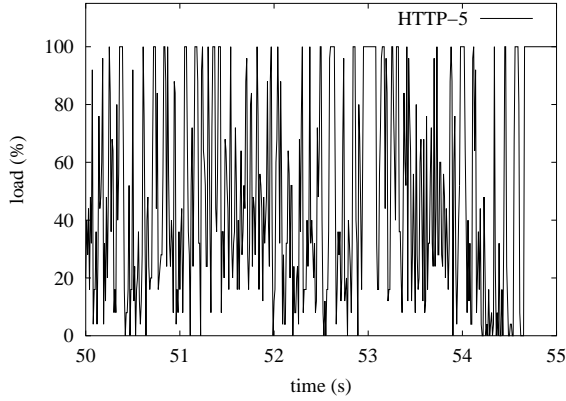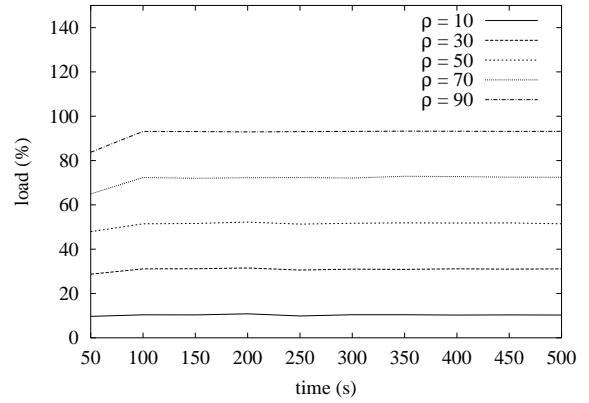
(a) Mean load variation under 10ms time interval.
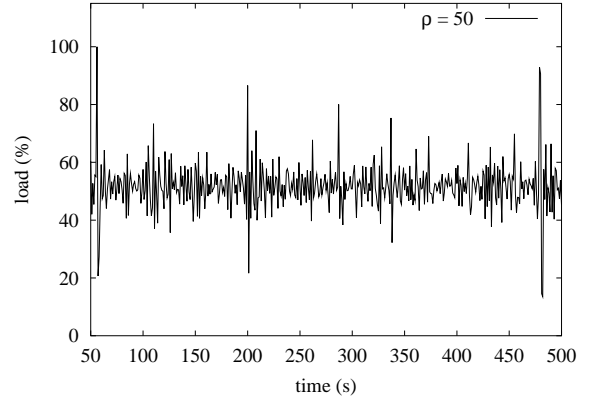
(b) Mean load variation under 1s time interval.
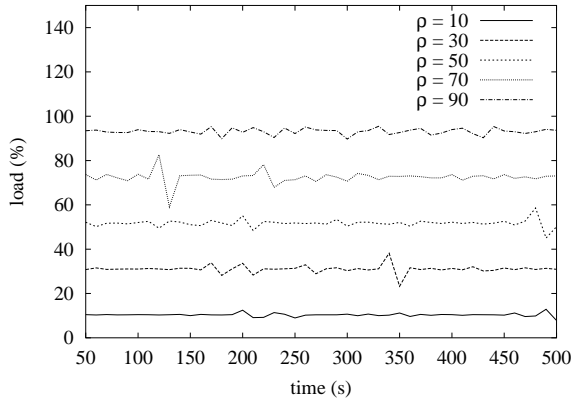
(c) Mean load variation under 10s time interval.

(d) Mean load variation under 50s time interval.

Figure 5: Mean load variation.

3.3 Self-similarity

In computer networks, asymptotically second-order self-similarity can be summarized as the property of having observable bursts on several (or all) time scales. Self-similarity is mainly evaluated by the Hurst parameter (H), which is described in the following interval: $0.5 < H < 1$. As $H \to 1$, the degree of self-similarity increases.

Some works have highlighted the existence and consequences of self-similarity in web traffic [15, 16]. The interest on self-similar process arises due to the consequences on network behavior. It has been shown that self-similarity can affect, in some extent, the buffers of network components and then increase loss rate.

In this context, it is important that an HTTP traffic model presents self-similarity if an experiment demands. The proposed model was evaluated and and sample results are presented in figures 9(a) and 9(b). The Hurst parameter was measured by the wavelet estimator introduced in [20] with minor modification to exhibit H as part of the graphic's title. The figures show that the model is able to reproduce different values of self-similarity. Figure 9(b) present an example of strong self-similarity ($H = 0.889$) with bursts varying from a few to hundreds of seconds, i.e. two orders of magnitude.
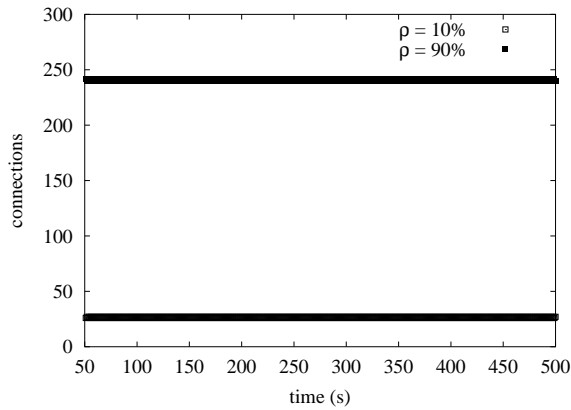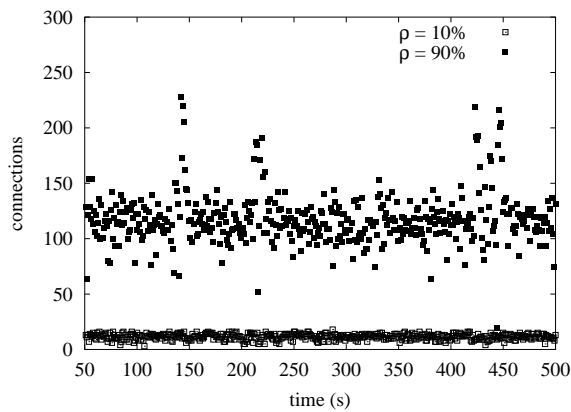
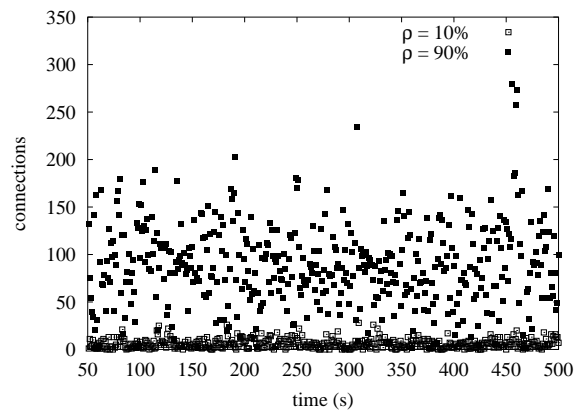(a) Fixed transfer size of 1K.
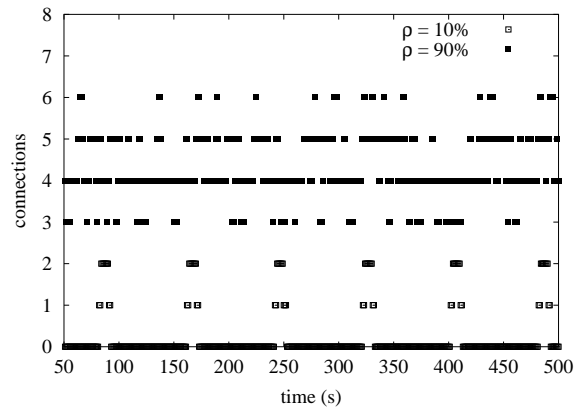
(b) Fixed transfer size of 5M.

Figure 6: Number of simultaneous connections of fixed transfer size.

(a) HTTP-1.

(b) HTTP-3.

Figure 7: Number of simultaneous connections of HTTP transfer.

4 Model Application

Since the aggregation model presented in this paper was intended to be used as input to bottle-neck links, some specific uses are adequate. Bottleneck links are basically routers, switches or similar equipments. In this kind of network elements, some mechanisms, policies and disciplines are of interest. Examples of these are:

- queue management, e.g. RED, BLUE, REM, etc.;

- schedulers, e.g. WFQ, GPS, etc.;

- markers, shapers, etc.

The model was developed in such way that is easy to generate traffic for one or several classes under a unique load control. This is useful in evaluation of quality of service architectures such as DiffServ.
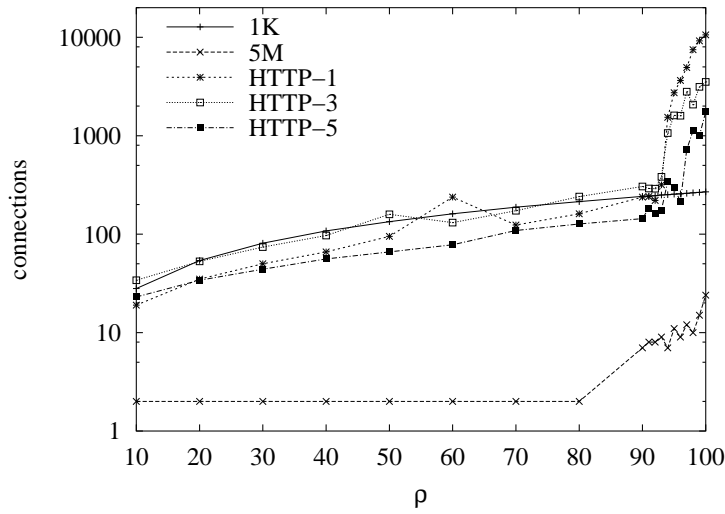
Figure 8: Number of simultaneous connections under different loads.
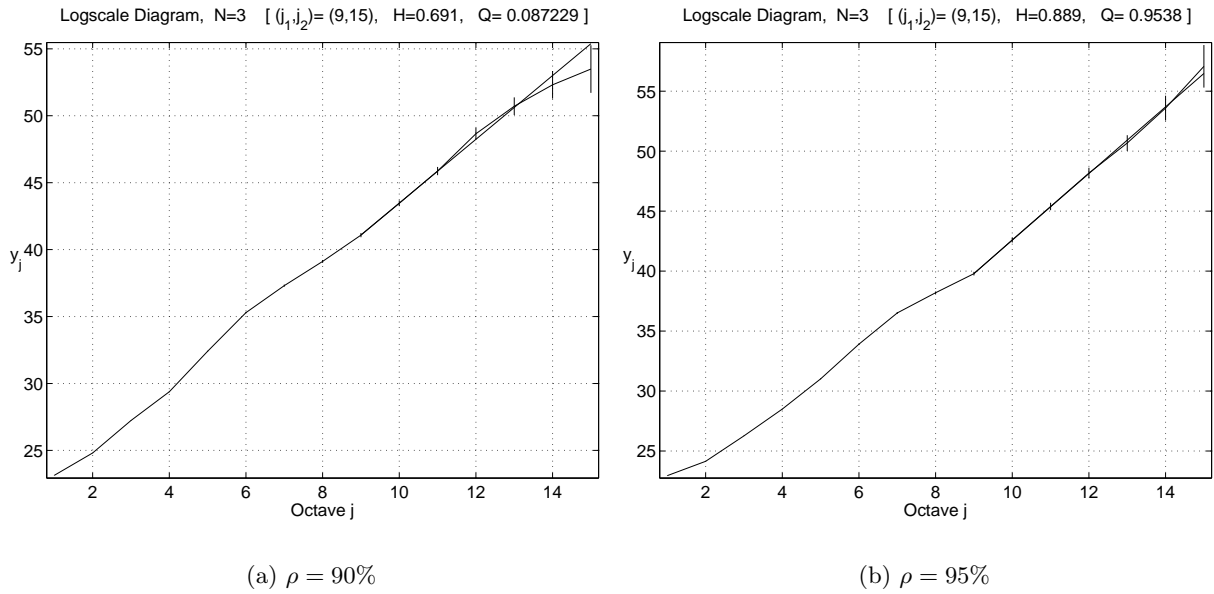

(a) $\rho = 90\%$

(b) $\rho = 95\%$

Figure 9: Wavelet analysis of cumulative work process of HTTP-1.

The model can also be used to evaluate congestion control algorithms since it present simple controls for load, transfer sizes and number of connections. In addition, the model can be applied as background traffic, which helps the evaluation of the influence of web traffic on other kinds of traffic.

Figure 10 shows an example of the model application. The simulation intended to evaluate the effectiveness of selective discard mechanisms for HTTP transfers. In order to assess the sensitivity of the discarding mechanisms, class 1 load was kept constant and the total load was increased up to 90%. A desirable result would be the remaining of class 1 to keep performance constant with the increase of the total load. As the figure shows, the average bandwidth per class when ρ_1 is fixed at 20%. Simulation experiments with different values of ρ, ρ_1 and ρ_2 were run,

and similar results were observed. As can be seen in figure 10, class 1 obtained bandwidth does not significantly change under PRIO (Push-out RIO) and RIO (RED with In/Out bit) policies. It is worth noting that the joint use of push-out and RIO does not offer any improvement to both classes. PO (Push-Out) is the most sensitive mechanism to the load increase, and it does not significantly differ the priority classes. Detailed comments about this experiment and some other uses of the model can be viewed in [21].
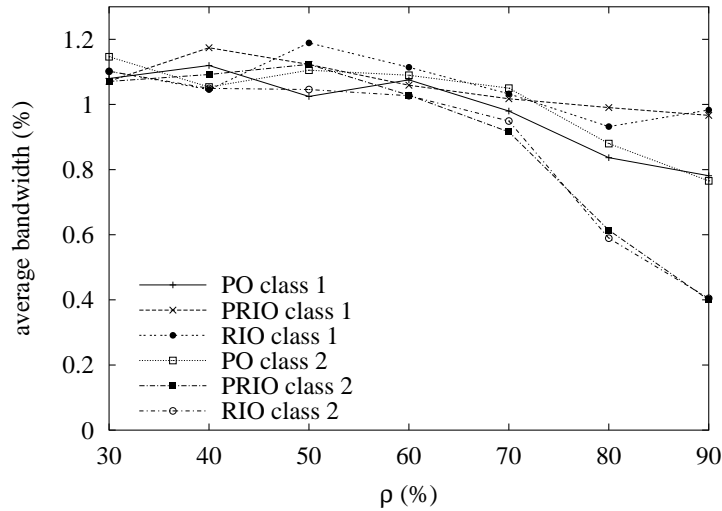

Figure 10: Bandwidth of classes 1 and 2 as a function of network load.

5 Conclusion

In this paper, a new HTTP aggregation model was proposed. The development steps were detailed and the model evaluation shown its benefits and shortcomings. The traffic model presented the ability to reproduce some important HTTP characteristics, which include transfer sizes and self-similarity. To control network load in an easy and precise manner is the main model feature. Examples of how to use the traffic model are also illustrated. As future plans to the traffic model, there are an under-development implementation based on sockets to be used in real network environments, evaluation of new mechanisms and model improvements.

References

[1] "CAIDA (Cooperative Association for Internet Data Analysis) - Characterization of Internet traffic loads, segregated by application." http://www.caida.org/analysis/workload/byapplication/.

[2] B. Mah, "An empirical model of http network traffic," in *Proc. INFOCOM'97*, Apr. 1997.

[3] M. Molina, P. Castelli, and G. Foddis, "Web Traffic Modeling Exploiting TCP Connections' Temporal Clustering through HTML-REDUCE," *IEEE Network Magazine*, vol. 14, no. 3, pp. 46–55, 2000.

[4] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *Proc. ACM SIGMETRICS Conference*, pp. 151–160, July 1998.

[5] H. Abrahamsson and B. Ahlgren, "Using Empirical Distributions to Characterize Web Client Traffic and to Generate Synthetic Traffic," in *IEEE/Globecomm'00*, (San Francisco), Nov. 2000.

[6] H.-K. Choi and J. O. Limb, "A Behavioural Model of Web Traffic," in *International Conference of Networking Protocol 99 (ICNP99)*, 1999.

[7] S. Deng, "Empirical model of WWW document arrivals at access link," in *ICC - International Communication Conference*, 1996.

[8] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol – HTTP/1.0," *Internet RFC 1945*, May 1996.

[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," *Internet RFC 2616*, Apr. 1999.

[10] E. Casilari, A. Reyes, F. J. Gonzalez, A. D. Estrella, and F. Sandoval, "Characterisation of Web Traffic," in *Internet Performance Symposium*, (San Antonio), Nov. 2001.

[11] L. D. Catledge and J. E. Pitkow, "Characterizing browsing strategies in the World-Wide Web," in *Third International World Wide Web Conference*, (San Antonio), Apr. 1995.

[12] C. R. Cunha, A. Bestavros, and M. E. Crovella, "Characteristics of WWW client-based traces," tech. rep., Boston University, July 1995.

[13] A. Feldmann, "BLT: Bi-layer tracing of HTTP and TCP/IP," *WWW9 / Computer Networks*, vol. 33, no. 1-6, pp. 321–335, 2000.

[14] L. Kleinrock, *Queueing Systems*. ISBN 0471491101, John Wiley and Sons Inc., 1975.

[15] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and possible causes," in *Proc. of the ACM SIGMETRICS Conference on Measurement & Modeling of Computer Systems*, (Philadelphia), pp. 160–169, 1996.

[16] K. Park, G. Kim, and M. Crovella, "On the relationship between file sizes, transport protocols, and self-similar network traffic," in *Proc. IEEE International Conference on Network Protocols*, pp. 171–180, Oct. 1996.

[17] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in Web Client Access patterns: Characteristics and Caching Implications," in *Special Issue on Characterization and Performance Evaluation*, 1999.

[18] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, "Tunning RED for Web Traffic," in *Proc. ACM/SIGCOMM'00*, (Stockholm), 2000.

[19] "IRTF end2end-interest mailing list archive." ftp://ftp.isi.edu/end2end/end2end-interest-1998.mail.

[20] D. Veitch and P. Abry, "A Wavelet Based Joint Estimator of the Parameters of Long-Range Dependence," in *IEEE Trans. on Info. Theory, Special Issue on Multiscale Statistical Signal Analysis and its applications*, Apr. 1999.

[21] K. V. Cardoso, J. F. de Rezende, and N. L. S. Fonseca, "On the Effectiveness of Push-out Mechanisms for the Discard of TCP Packets," in *IEEE International Conference on Communications*, Apr. 2002.