Estimating the Service Time of Web Clients using Server Logs

Oscar Ardaiz
Dep. Arquitectura de Computadores
Universidad Politécnica de Catalunya
Jordi Girona, 1-3, Barcelona (España)
oardaiz@ac.upc.es

Felix Freitag
Dep. Arquitectura de Computadores
Universidad Politécnica de Catalunya
Jordi Girona, 1-3, Barcelona (España)
felix@ac.upc.es

Leandro Navarro
Dep. Arquitectura de Computadores
Universidad Politécnica de Catalunya
Jordi Girona, 1-3, Barcelona (España)
leandro@ac.upc.es

SUMMARY

This article proposes and evaluates measures for estimating the service time of a web client using server logs, only from the server side without introducing traffic into the network. The HTTP protocol is described as well as the different interactions between the web server, the communication components, and the web client application. The first measure is based on the time it takes for the web server application to deliver an object to its operating system, keeping in mind the buffer effect of the server network. The second measure also considers the inter-arrival times to the server application of the GET requests for the objects that are part of a web page. We propose formulas, validated experimentally, that relate the proposed measurements in the server with the different components that take part in a web transaction and the service time experienced by clients. We have carried out several experiments to evaluate the validity of the proposed measurements and the best measure estimates the service time of the client with an error below 20% for 90% of the requests. We observed a cyclic component in the measurements of the server that can simplify the estimation of future values. For this reason, the proposed measures may be used to know what is the service time the client perceives in each visit to a server. It can also be used by content distribution networks to choose between several replica located in different places in the Internet.

Keywords:

Web, Metrics, Server Selection

1. INTRODUCTION

A metric that defines the performance of a web application is the total connection time experienced by the client or user service time: from the moment the user requests a document until it receives the last character of the last component of the page (a web document can be made up of several html objects or graphics). However, this metric depends on the traffic of the network, the load of the server, and the size of the document. The main problem with this metric is that the measurement must be done at the client's site. It would be desirable that a metric used by a service provider to make a decision, like selecting dynamically the best server for a client within a group, should be obtained at the server location itself, and with a lower cost. It is possible to obtain measurements by capturing network traffic at the server; however, these measurements are expensive. It would be desirable to obtain these measurements at the application layer, in order to minimize the modifications introduced, without adding additional traffic to the network and without excessive additional cost.

The goal of this paper is to propose and evaluate measurements obtained in a web server in order to estimate the performance perceived by clients when accessing it.

2. RELATED WORK

To obtain metrics in a simple way, which should permit the comparison of service time that can be obtained from different web servers, is a need of several Internet services. The main feature of these services is that they have different servers distributed throughout the Internet, allowing a client to access the service through any of them, albeit with different service qualities, depending on their position relative to clients or servers. In these services it is necessary to select the server that can provide the best service time for a certain client. Services such as Content Distribution Networks described in [24] and [5], already marketed by different companies, such as Akamai [1], need an infrastructure that measures, compares, and offers the location of the best server for a certain client at a given moment.

There are different proposals of infrastructures that measure and offer the best location of a server or the values of the measures for each one of the eligible servers. Pinger [8] is an infrastructure that analyzes the behavior of Internet and offers measurements at network layer that help to estimate the performance that can be experienced by different applications. Idmaps [11] is a proposal to create a global network of elements that measure the distance between any two nodes in the Internet; Sonar [20] is a proposal to modify the DNS service with measurements of proximity between Internet points. And, in this area there are also commercial services such as KeyNote [15] that offer to evaluate the service time provided by the servers of the contracting company from different points in the network.

There exist several proposals of metrics to select which server will provide the best service among a group of candidates. There are three types of metrics: static, based on the configuration values of the network and servers; statistical, based on the measurements carried out during previous requests, which can be too out of date;

and dynamic, based on measurements carried out when the service has been requested, which increases service time and creates additional load. For both types of metric, statistical and dynamic, different values can be measured in order to construct the metric: the round trip time (RTT) of a data block, the number of links separating the client from the server, the end-to-end bandwidth between the client and the server, the load of the server, the response time of the server, etc.

The different proposals of metrics, classified in Table 1, make comparisons between one and another, and propose architectures to create this service; however, few of them compare their metric with the ideal metric, which is the client service time. Dykes et. al. [9] evaluates different metrics of service time of the client, but these are metrics that are taken only at the client.

Our objective is to obtain a metric at the server, the point that can be accessed by any service provider, as close as possible to the service time experienced by the client. Our intention is also that this metric may be obtained in the simplest manner, in the server itself without introducing additional traffic in the network.

Our metrics are obtained from measurements at the server. These measurements are kept in the log files created by web servers. There are some proposals aimed at modifying web server logs, since they do not contain time data with a resolution above normal, milliseconds or microseconds instead of seconds [6], or simply because they are too basic and require too many manipulations in order to obtain information about the users of the server [23].

Table 1. Classification and examples of different metrics.

Category	Metric	Example
Statistical / at client	Bandwidth	SPAND [26]
Dynamic / at client	RTT	Experimental Evaluation [22]
Dynamic / at client	Bandwidth, RTT, hops	bprobe [4], WAWM [3]
Dynamic / at client	RTT, latency, BW	Experimental Evaluation [9]
Statistical / at server	Routing metrics	IBM Distributed Director[14]
Statistical / at server	Bandwidth	Network Whether Service [10]
Dynamic / at server	Load and others	Akamai Inc [1]

3. ANALYSIS OF METRICS

We review briefly the HTTP, TCP protocols and their interactions. We shall analyze what metrics can be obtained in a web server, and finally we will study the metrics we are proposing.

3.1 HTTP and TCP Protocols

The HTTP and TCP protocols, as well as their interactions, have been studied widely in different works. The HTTP protocol, in its Version 1.0, is a very simple protocol in which a TCP connection is opened, a GET request is sent and a response is received in one or several TCP segments. Its Version 1.1 makes it possible that once this TCP connection is open, several GET requests can be sent at the same time and several objects can be received using the same connection, or that the requests be made once the previous object has been received. For this reason the performance of HTTP depends to a great extent on the performance of the TCP function as can be seen in Figure 1.

The transfer rate in the TCP protocol depends on several mechanisms that this protocol implements to exercise congestion control and flow control. TCP uses a mechanism called slow start with which it attempts to avoid congestion in the network. The effect it has on the transfer rate grows exponentially as data is transferred. Flow control is carried out by means of a window type mechanism of which there are several versions, and this basically means that n data blocks can be transmitted without having obtained confirmation of reception from the server. The problem is to determine the size of the window. For bandwidth products with high round trip times (RTT) it is possible to have sent several TCP segments without receiving confirmation.

Consequently, the transfer rate of TCP and HTTP depends on the traffic in the network, which means the effective bandwidth for data transmission decreases as traffic increases, and causes the round trip time to vary in line with the accumulated delay in the queues of the intermediate nodes according to the existing traffic. In [7] there is a detailed analysis of performance of TCP in relation to network characteristics.

The interactions between the HTTP and TCP protocols, together with their possible improvements, have been studied, for example in [13] where the conclusion is that either TCP connections are reused, as in the case of HTTP 1.1 or the TCP connection parameters are cached. [18] carries out an empirical study of the relationship between RTT values and response times of HTTP.

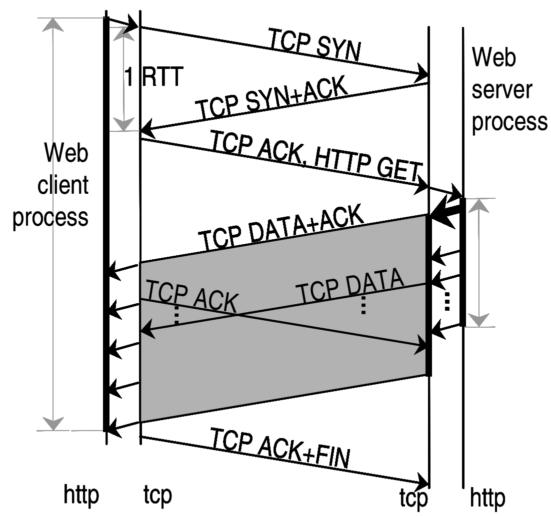

Figure 1. HTTP-TCP interactions.

3.2 Web Server Logs

A log line in CLF ("Common Log Format") format of a web server contains these fields: name of client's machine, two fields for client authentication, date and hour of completion of request, request received, response status, size of object returned:

a.x.es – [16/May/2000:13:26:25+0200]"GET/manual/index.htmlHTTP/1.0" 200 2874

It is obvious that the information that this log provides about service time of a client is insufficient. We found four major deficiencies:

- it does not provide information on the duration of the transference of the response,
- the time resolution is in seconds, when we are needing a precision in milliseconds,
- the number of bytes transferred is related to the size of the object, but does not take into account the transmission of the HTTP header,
- the DNS resolution of the name of the client is done before marking the final time of the request, and this often adds a very significant time.

The modifications that have been done in order to obtain the measurements we need are these: web server instrumentation so that it keeps the time of initiation and termination of each transaction in milliseconds, obtaining the size of the HTTP header sent, and not doing the reverse IP DNS resolution.

There is a buffer between the web server and the TCP socket that has the purpose of queuing the data that the server generates in order to send it through the network. The measured time of completion of the request corresponds to the last write operation in the buffer, The buffer can be so large that the response can arrive significantly later to the customer. The smaller the buffer, the less memory is used; however, if the buffer is very small there can be occasions in which the transmission speed worsens because it is impossible to transmit the quantity of data that the network is capable of accepting. [12] explains that in those situations in which the bandwidth multiplied by the round trip time between the web client and the server is very large, the TCP window can open considerably which allows a great amount of data to be in transit without confirmation. If the buffer is not very big, it will not be possible to have as much data while waiting for a confirmation with a consequent waste of bandwidth. In [25] experiments are performed with buffer sizes that adapt to end-to-end link capacity, and memory usage are analyzed, which in the majority of cases are no inconvenience compared with the improvement in performance. As we shall see in Section 3.3, we shall calculate the time to empty the buffer from the times obtained in the logs, with known buffer sizes.

With these modifications, the new fields in positions 4, 5, and 9 of the log are: time of reception of request from the server in microseconds, time of completion of writing in the transmission buffer, and the size of the HTTP header in the response:

144.33.22.11 – 166666.234542 166666.435123 "GET/manual/index.html HTTP/1.0" 200 2874 212

We know of two log analysis that have been done after instrumenting a web server in which measurements were obtained like those we are looking for. Manley et. al. [17] measured what proportion of a web transaction is due to processing the request, searching the disk, and transmission to the network. They concluded that in overloaded servers the processing time of the request could be significant. Mogul [19] measured in an overloaded server the correlation between the size of an object and the time logged by the server, concluding that there ceased to be any correlation below 30 Kbytes. As we will see in the analysis and later experiments, this probably happened due to the existence of a buffer whose size was close to that value.

3.3 Proposed Metrics and Time Perceived by the Client

The time measurements we will be performing on the server are seen in Figures 2 and 3:

T_A= time of the last writing in the buffer by the web server in a valid response to reception of GET method – time in which the reception of a GET request is completed by a web server.

T_{AB}= time in which the reception of a GET request is completed for an embedded object – time in which the reception of a GET request is completed for an html document by the web server.

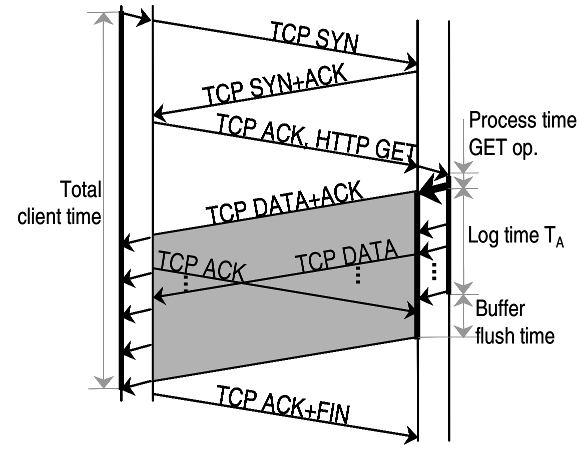

Figure 2. T_A in relation to $T^{cliente}_{total(html)}$

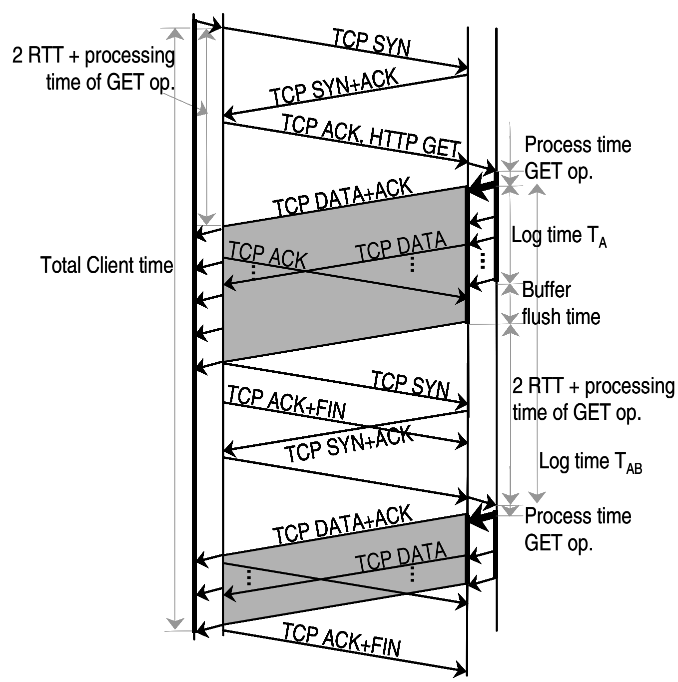

Figure 3. T_{AB} in relation to $T^{cliente}_{total(html+gráfico)}$

The second measurement can be done when a web page in html format has a graph inserted with a <img> tag. In this case the web client, as soon as it perceives that the document has an image, opens a new TCP connection and starts to receive this image at the same time. Thus the web client can visualize the graphics at the same time as it continues to receive the rest of the html document. This occurs in http 1.0 as well as http 1.1. In Figure 3 we show the case of an html document that contains at the end a single reference to an image. The measurement is done using 2 different lines of the log, whose correspondence must be obtained when the <img> relation of inclusion between both objects is detected in the html object.

We also need to know the buffer sizes, documents transmitted, headers transmitted, and the position in bytes of the <img> tag in the document.

B = buffer size

X_h = (transferred bytes) html document size + header sizes

X_g = (transferred bytes) size of graphic document + header sizes

X_a = transmitted bytes until <img> reference = document size until <img> + header sizes

T_A represents the transfer time of X_h – B bytes, when the end of the last write operation is logged, B bytes are still waiting at the buffer to be transmitted.

The first metric we propose represents the transference time of a byte, that is the inverse of the average end-to-end bandwidth:

$$m_1 = \frac{T_A}{X_h - B}$$

The second metric that we propose is the one obtained by dividing TAB into the number of bytes transmitted up to the <img> reference:

$$m_2 = \frac{T_{AB}}{X_a}$$

In Figure 3 we can see that TAB is equal to:

$$T_{AB} = m_1 \cdot X_a + 2RTT + T^{servidor}_{procesoGET} + T^{cliente}_{proceso}$$

therefore:

$$m_2 = \frac{T_{AB}}{X_a} = m_1 + \frac{2RTT + T^{servidor}_{procesoGET} + T^{cliente}_{proceso}}{X_a}$$

This means that m2 depends on the round trip time (RTT), the processing time of the server, the processing time of the client, and inversely on the byte size of the fragment transferred up to the reference.

Metric m_2 differs from m1 in that it considers the available bandwidth and other factors that affect the total time that the client observes. Moreover, m2 does not depend on the size of the buffer.

In Figure 2 we can see that the total time of the transaction that the client experiences as a function of m_1 is, for a single html document:

$$T^{cliente}_{total(html)} = m_1 \cdot X_h + 2RTT + T^{servidor}_{procesoGET}$$

And from Figure 3, in which $X_a = X_b$, the total transaction time of the document and the embedded image is:

$$T^{cliente}_{total(html+gráfico)} = m_2 \cdot (X_a + X_g) + \left(1 - \frac{X_a}{X_g}\right) \cdot \left(2RTT + T^{servidor}_{procesoGET}\right) - \frac{X_a}{X_g} T^{cliente}_{proceso}$$

From m1 and m2, we define the estimators T_{e1} and T_{e2} of service time experienced by the client T^{client}_{total} as a function of measurements at the server:

$$T_{e1} = m_1 \cdot X$$
$$T_{e2} = m_2 \cdot X$$

where X is the size in bytes of a document made up of one or more objects.

4. EXPERIMENTATION

To verify the previous model, we have modified the Apache web server (http://www.apache.org/httpd.html version 1.3.12) so that it can take time measurements with a high resolution. Measurements have been taken in microseconds using the gettimeofday call of Unix, since Linux and Solaris are the operating systems of the servers used in the experiment, although the same function or a similar one exists for many other operating systems.

Once the sever receives a request from the client (generally GET) a timestamp is made and kept in the memory "pool" of the request. Once the server has finished writing the object to the buffer, a second timestamp is made. The module mod_1og in the Apache server has been modified also to allow for the generation of log lines as proposed in Section 3.2. With a test program, we have verified that the calls to gettimeofday do not introduce a significant delay.

As a client, we used the webbot program, published by the W3C consortium, that allows the simulation of the behavior of a graphic web client (one that can load an html document and associated images). In addition, we used the lynx client program since it is preinstalled in Linux.

Measurements have been taken from two servers: S1 is a Linux (Red Hat 6.1) machine connected to an ISP by a 256 Kbit/s line with a very high traffic load, and S2 is a Solaris (Solaris 2.6) machine connected to a academic network, both located in Barcelona (Spain). In these machines, various web servers have been installed with different buffer sizes (4 Kb, 50 Kb, 100 Kb, and the default value).

We have placed 4 clients in different locations: C1 is a host situated 4 network hops away from S2, connected to the network by a 256 Kbits/s line with a great traffic load; C2 is a host situated 2 hops away from S1 and 8 hops away from S2 crossing the university network backbone and a national traffic exchange, both located in Barcelona. We have also used C3, a Linux host located in an ISP in Buenos Aires (Argentina),12 network hops away from the servers, with a high capacity link and high delay (high bandwidth delay product), as well as C4, a Linux host with limited link capacity, high occupancy and high delay in Santa Fe de Bogotá (Colombia),19 network hops away from the servers.

To evaluate the proposed metrics two experiments have been done:

- E1: The objectives are: a) to observe the influence of the web server buffer on the measurements made at the server, for different buffer sizes, and b) to compare the measurement that appears in the server log (T_A) with the measurement at the client,($T_{total\ client}$) for various object sizes.

  Requests were made between C3 and S1, C2 and S1, C1 and S2 , for an object with sizes ranging from 1 Kb to 500 Kb (1 Kb steps between 1-10 Kbytes, 10 Kb steps between 10 Kb and 100 Kb, 100 Kb steps between 100 Kb and 500 Kb), to servers with different buffer sizes (4 Kb, 50 Kb, and 100 Kb).

- E2: The objective is to evaluate, during a long period of time, the metrics proposed for estimating the service time of the client.

  Requests were made between C1 and S1, C3 and S1, C3 and S2, for a web document of 70 Kb made up of an html file of 50 Kb and a gif graphic of 20 Kb, every 5 minutes during 10 days running. The html file ends with a “<img src>” reference to the graphic so that $X_a = X_h$. The server has been configured with a 4 Kb buffer.

We measured the header sizes and we have proved that they are negligible compared to the size of objects to be transferred. In addition, the processing time of the client and the server have been calculated, so that in normal load circumstances they can be discarded compared to the rest of the time measurements.

5. RESULTS AND DISCUSSION

In this section we show the results of Experiments E1 and E2.

Figures 4 and 5 display client (T_{client}) and server (T_A) measurements of Experiment E1. The three runs of the experiment show equivalent results. We show the measurements between C3 and S1. Six requests were executed for each object and we calculated the median of the values measured (the central value of each group of measurements).

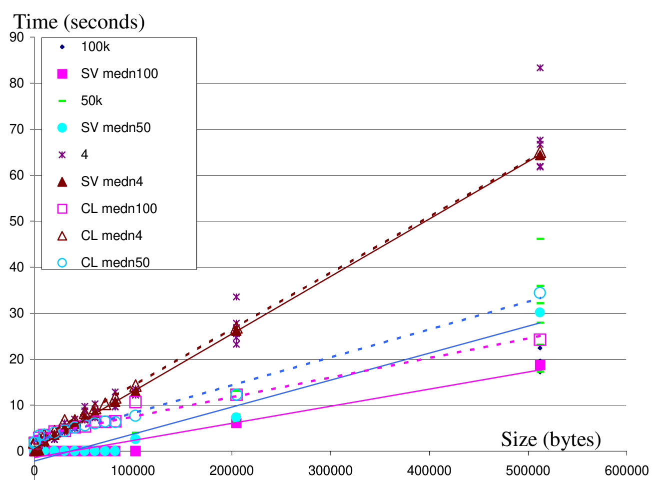

Figure 4. Values and tendencies of TA and Tclient (secs) vs. size of object and buffer (bytes). Legend: Median of client/server buffer size.

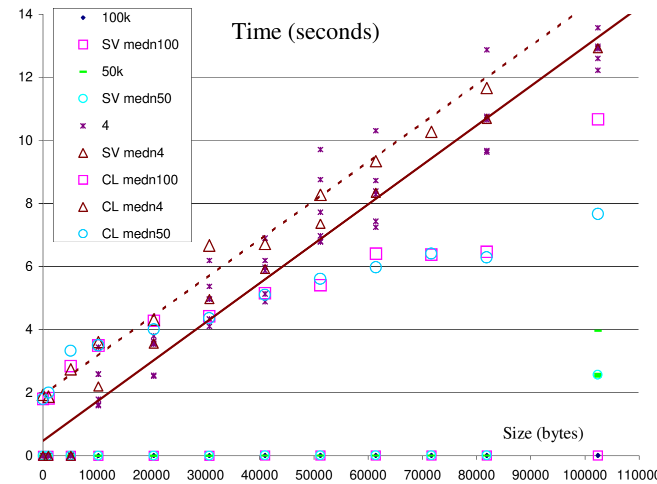

Figure 5. Idem figure 4 for object sizes <=100Kb.

The dotted lines indicate the tendency of the median values measured at the client and the solid line the tendency of median values measured at the server. Figure 5 shows that for objects smaller than the size of the buffer, the measurement at the server is almost nil since the server writes the complete object in a single write operation without suffering any effects from the network. For objects greater than the buffer, both lines have almost the same slope and values are relatively close. Therefore, the measurement at the server is a good approximation of the measurement at the client, which supports the validity of the proposed metrics.

The results of Experiment 2 are shown in figures 6, 7, 8, and 9.

In Figure 6 we compare the measurements taken at the client T_{c1} (top line), the estimation of T_{e2} based on the metric m_2 (intermediate line), and the estimation of T_{e1} based on the metric m_1 (bottom line), measured every 5 minutes during a 4 hour period with high traffic (part of the 10 day logs of Experiment 2, shown in Figure 8) between C1 and S1.

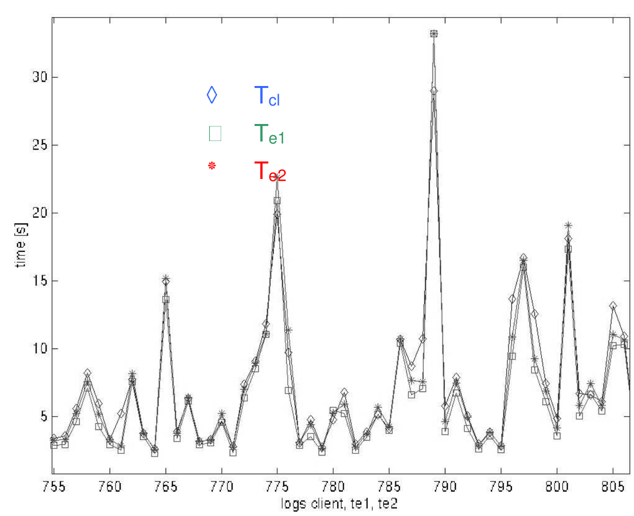

Figure 6. Tclient Te1 and Te2 values during several hours

The value of T_{e1} is always lower than Tclient because it does not take into account the RTT, while T_{e2} does include the RTT effect. We can see that even in this high load situation, the values of T_{e1} and T_{e2} calculated at the server follow quite closely the evolution of what the client perceives.

As to the precision of measurement, we calculated the CDF function of the T_{e1} and T_{e2} estimations compared to the measurement at the client. This was calculated compared to all the measurements of Experiment E2 , every 5 minutes during a 10-day period. Figure 7 shows the CDF of experiment E2 between C1 and S1. We can see that the line corresponding to Te2 is always above the one for Te1. These values indicate the percentage of measurements whose relative error $|(T_{client} - T_e)/$ Tclient$|$ is below a given value. We can also see that the estimation is quite accurate since the relative error of the T_{client} estimation is less than 20% for 75% of values of Te1 and for 90% of values of Te2.

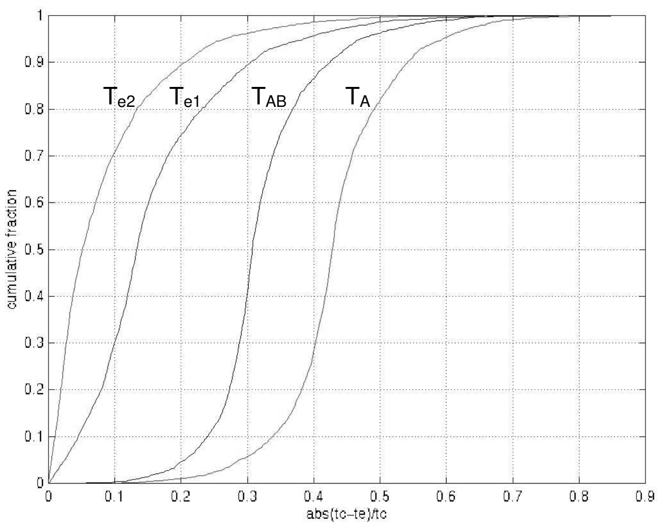

Figure 7. CDF relative error T_{e1},T_{e2},T_A,T_{AB} **compared to** T_{client}

We believe that the estimations are valid for use in several applications such as selecting one server among several. As it is discussed in the IDmaps [11] proposal, the important thing is the relative error that can appear when selecting a server over another; thus a difference of 20% in the estimated values for two servers would mean that the selection of a server is unimportant since the difference in service time that the client experiences accessing one or the other will not be great. However, in those cases in which the difference in service time the client would obtain in a server compared to another one were of several orders of magnitude, then it becomes important to make a good selection. Our metric would behave well in these cases, since the error in estimation will be of several orders of magnitude in very few cases. [16] also comments that the commercial content distribution networks (CDN) usually are satisfactory if a client avoids picking the worst server, since traffic conditions can vary rapidly.

Another useful aspect of a measurement at the server is that it enables us to predict values. Figure 8 shows service time measured at the client during 10 days. We can see clearly the cyclic pattern following the daily and weekly work cycles of the region where the server is located (peaks between 9 am –2 p.m. 3 p.m. – 6 p.m., and to a lesser extent from 10 p.m. – 12 p.m.). These graphs can vary in other servers depending on the server under study, its usage and its content.

From the logs in Experiment 2 between C1 and S1 that are shown in Figure 8, we have calculated the autocorrelation for detecting periodicity in the server logs, which one can use to apply predictive techniques.

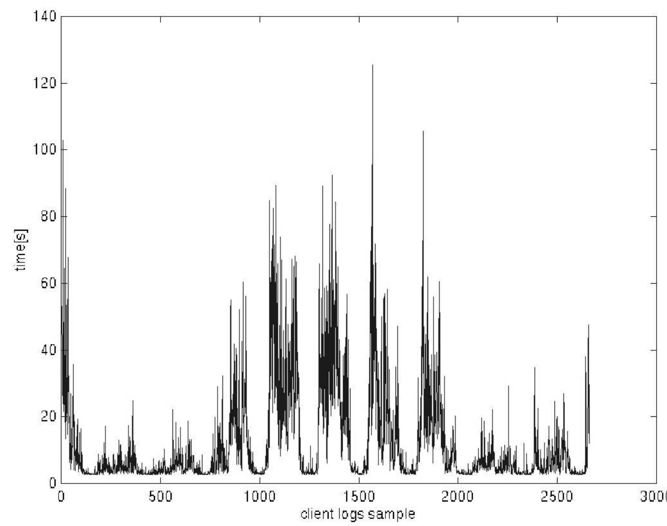

Figure 8. Tclient values every 5 minutes during 10 days.

Figure 9 shows the autocorrelation of the server logs and the client. We can see that the peaks coincide, appearing with a periodicity of one day. In addition, a peak is observed between 1900 and 2000, which indicates the periodicity in one week. The value of the peaks in the autocorrelation function can be observed graphically but it is not very large. This indicates that there is certain dominant periodicity in the logs, but there is also a part that is not periodic.

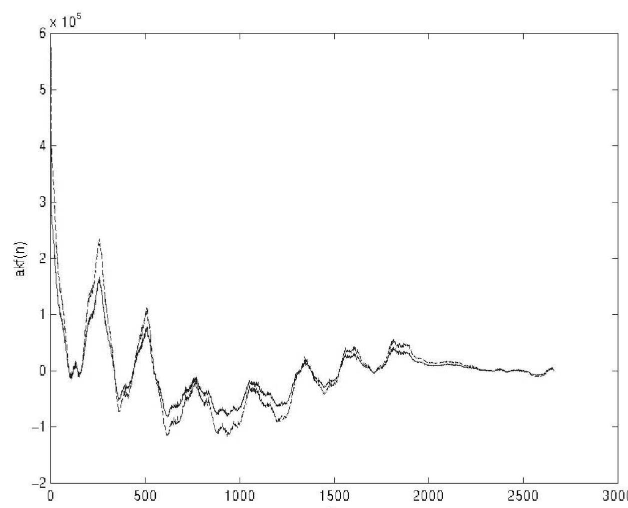

Figure 9. Autocorrelation function of values in Figure 8.

The previous result permits us to indicate that some historical measures can help to calculate estimations of future values, which is a promising result for designing predictors of future behavior of a server from the metrics analyzed here. Nonetheless, the predictability of the measures depends on many different factors for which reason it is necessary to carry out a more exhaustive study to better characterize in which situations future values can or cannot be predicted. We have seen that some measurements have a fractal behavior, meaning that sometimes it is nearly impossible to predict the future behavior of a series of values.

Other studies that evaluate statistical estimators have found that there is a certain stability in the measurement of web performance regarding spatial localization and regarding time in short intervals [2]. From these we can deduce that future values can be estimated weighing with greater coefficients the most recent measurements as in topology-d [21], or doing regression analysis like in Network Whether Service [10]. Our first results in this regard indicate that such stability exists and that we can search for some autocorrelation coefficients with which an attempt to predict the response times of a client can be done.

6. ACKNOWLEDGEMENTS

This study was partially financed by the CICYT-MCYT. Our gratitude to Narcís Vives de Espais Telematics S. L. (Spain), to Claudio Righetti from Prima (Argentina), Joseph Turró from Eucanex (Ecuador), and Julián Casasbuenas from Colnodo (Colombia) for the permission to carry out tests from their machines and networks.

7. CONCLUSIONS

We have shown that with measurements taken from the server and without introducing traffic on the network, it is possible to estimate the service time experienced by a web client.

We have analyzed the metrics that can be obtained at a web server regarding the service time experienced by a client, which has components that depend on the round trip time and bandwidth, difficult to obtain without modifications in all web browsers. We have compared these metrics in an experiment with clients and servers situated in different locations in the Internet, and therefore under the same conditions of variability and network load as any normal client would experience.

We have proved experimentally how the size of the buffer influences the measurements that are obtained at the server, as we had analyzed. The experimental results corroborate our analytical hypotheses regarding the degree of estimation of service time at the client that can be obtained with both metrics.

We obtained a very accurate estimation, with a relative error in T_{client} of less than 20% for 75% of values of Te1 and for 90% of values of Te2.

The modifications which we made in the server, apart from being useful in getting measurements with which to be able to choose among several servers, can be helpful for web server administrators in order to know what is the real service time experienced by visitors accessing their servers so as to act upon this information and redesign its contents.

8. REFERENCES

[1] Akamai Inc., “Freeflow”, Dic 1999, http://www.akamai.com/

[2] H.Balakrishman, M. Stemm, S. Seshan, R. Katz “Analyzing Stability in Wide-Area Network performance” Proceedings from “1997 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems”. 1997, Seattle, WA

[3] P.Barford and M. E. Crovella “Measuring Web Performance in the Wide Area”, Performance Evaluation Review, August 1999

[4] R.L. Carter, M.E. Crovella, “Dynamic Server Selection using Bandwidth Probing in wide area Networks” Tech. Rep. BU-CS-96-007. CS Dept. Boston University., March 1996.

[5] J. Chase, M. Rabinovich, “Web Caching and Content Distribution: A view from the Interior” , 5th Web Caching Workshop, Lisbon, May 2000.

[6] I. Cooper, J. Dilley, “Known HTTP Proxy/Caching Problems” IETF draft draft- ietf-wrec-known-prob-02.txt, July 2000.

[7] L. Cotrell, J. Halperin, “Effects of Internet Performance on Web Response Times”, January 1997. http://.slac.stanford.edu/comp/net/wan-mon/ping/correlation.html

[8] L. Cotrell, “PingER project at Stanford”, December 1999 http://.slac.stanford.edu/comp/net/wan-mon/

[9] S. Dykes, K Robbins, C. Jeffery “An Empirical Evaluation of Client side Server Selection Algorithms”, Proc. of IEEE Infocom’00, March 2000

[10} M. Faerman, A. Su, R. Wolski, F. Berman “Adaptive Performance Prediction for Distributed Data-Intensive Applications” Super Computing99, Portland, 1999.

[11] P. Francis, S. Jasmin, V. Paxson, L. Zhang, D.F. Gryniewicz, Y. Jin “An Architecture for a Global Internet Host Distance Estimation Service”, Proceedings INFOCOM99, 1999.

[12] J. Heidemann, “Performance Interactions Between P-HTTP and TCP Implementations”, ACM Computer Communications Review, April 1997.

[13] J. Heidemann, K. Obraczka, J. Touch “Modeling the Performance of HTTP Over Several Transport Protocols” IEEE/ACM Transactions on Networking, V5, N5, October 1997.

[14] “IBM Distributed Director”, http://www.ibm.com, 1999

[15] KeyNote Systems Inc., “KeyNote Perspective”, http://www.KeyNote.com, December 1999.

[16] K. Johnson, J. Carr, M. Day and F. Kaashoek “The Measured Performance of Content Distribution Networks”, 5th International Web Caching and Content Delivery Workshop, Lisbon, 2000.

[17] S. Manley, M. Seltzer, “Web Facts & Fantasy” Proceedings of the 1997 USENIX Symposium on Internet Technologies and Systems, Monterey, CA, 1997.

[18] M. Mathis, J. Semke, J. Mahdavi, T. Ott “The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm”, Computer Communications Review Vol 27, No. 3, July 1997

[19] Jeffrey C. Mogul, “Network Behavior of a Busy Web Server and its Clients” Compaq WRL Research Report 95/5, October 1995.

[20] K. Moore, J. Cox, S. Green, “Sonar – a network proximity service”, available at http://www.netlib.org/utk/projects/sonar/ February 1996.

[21] Katia Obraczka and Grig Gheorghtu “The performance of a Service for Network-Aware Applications”, Minutes of “ACM Sigmetrics” Conference SPDT98

[22] K. Obraczka, F. Silva “Looking at Network Latency for Server Proximity” USC/ISI Tech Report 99-714. 1999

[23] J. E. Pitkow “In search of Reliable Usage Data on WWW”. The Sixth International World Wide Web Conference, Santa Clara, USA. 1997.

[24] M. Rabinovich, I. Rabinovich, R. Rajaraman, A. Aggarwal, “A dynamic Object Replication and Migration Protocol for an Internet Hosting Service”. IEEE Int. Conf. on Distributed Computing Systems, Lisbon, 1999.

[25] J. Semke, J. Mahdavi, M. Mathis “Automatic TCP Buffer Tuning” Computer Communications Review, Vol. 28, No 4, October 1998.

[26] M. Stemm, R. Katz, S. Seshan, “ A Network Measurement Architecture for Adaptive Applications”, Proc. of IEEE Infocom’00, March 2000.