

# Disassembling Web Site Response Time

Thiam Kian Chiew and Karen Renaud  
Department of Computing Science  
University of Glasgow  
17, Lilybank Gardens  
Glasgow G12 8RZ  
United Kingdom  
{tkchiew; karen}@dcs.gla.ac.uk

## Abstract

Web site latency can be measured by the response time to user requests. User perception of the responses is affected by psychological factors. This paper proposes a method which combines client-side measurement, server-side measurement, and active probing methods in order to measure response time for accessing a Web site as well as disassemble the time into the three constituents, namely transmission, processing/generation, and displaying/rendering. This paper also discusses some of the factors that contribute to delay in downloading Web pages.

## Keywords

Web, Measurement, Performance, Disassembling, Response Time, Constituent Times.

## 1. Introduction

The Web has become one of the most significant computer applications and the number of Internet users is growing exponentially. Both factors influence Web-based system performance requirements. As a result, performance issues for Web-based systems have been attracting much research interest in recent years.

There is no agreed metric for measuring the performance of computer systems, and even the definition of performance itself is open to debate (Lilja, 2000). In experimental computer science, analysis of performance can be thought of in terms of measurement, interpretation, and communication of a computer system's speed or capacity. This is the view adopted in this paper, where the concept of performance is taken to refer to the speed with which a Web site serves its users. The speed, termed *response time*, is defined as the time elapsed from the moment the user requests a Web page until the requested page is displayed (Marshak and

Levy, 2003). It is one of the major performance issues in the Web. Few studies have examined the users' tolerance of delay in downloading Web pages. (Dyson and Longshaw, 2004) report a user tolerance ranging from 3 to 20 seconds, and (Nielsen, 2000) believes 10 seconds to be the general cut-off point. However, such psychological factors are out with the scope of this paper.

Finding ways to attract more visitors to a Web site and encourage them to stay longer to use the services or buy the products, are goals that a Web site owner wishes to achieve. These goals will be hard to realise if the system has poor performance, with, for example, low response time to user requests (Debaraj et al., 2003). From a technical point of view, measuring response time of a Web site and understanding contributing factors to low response time are important tasks. It is worthwhile to be able to decompose response time into its constituting components in order to provide a detailed analysis of response time.

There are three basic categories for the methodologies used to measure response time: *client-side measurement*, *server-side measurement*, and *active probing*. This paper proposes a method which combines methods from these three categories in order to measure response time as well as to decompose response time into three of its constituents; *transmission*, *processing/generation*, and *displaying/rendering* as follows:

- *Transmission* consists of the time taken to transmit a request and response between the client and the server over the network.
- *Processing/Generation* is the portion of response time taken by the server to process the request and generate the response.
- *Displaying/Rendering* on the other hand refers to the time taken by the client machine to render and display the response to the user.

The proposed method can be used by a Web administrator to monitor the performance of particular pages in a Web-based system and identify those that deliver poor response times, enabling remedial actions to be taken.

This paper also discusses some of the factors that contribute to delays in delivering Web pages, which include the context of use, such as the number of Web browser windows concurrently being used and the users' browsing pattern in using the browser windows.

The sections of this paper is organised as follows. The three measurement methods mentioned above are described in Section 2. In Section 3, our methodology in conducting the study is explained. Data analysis and results are presented in Section 4. Section 5 concludes and discusses future work.

## **2. Measurement Methods**

This section describes three methods for measuring response time of a Web site: client-side measurement, server-side measurement, and active probing.

### **2.1 Client-side Measurement**

Client-side measurement makes use of instrumentations such as scripting languages or specialised software. Rajamony and Elnozahy (2001) used JavaScript to instrument a set of Web pages for measuring response time, whereby pages are referenced by instrumented links. When an instrumented link is activated, the current time is determined and remembered, and then the request is sent to the Web server. After the requested page and its embedded elements have been fully loaded to the browser, the client browser computes the response time as the difference between the current time and the previously stored time. The response

time can be transmitted to a record-keeping Web site on a different server from the originally responding Web server. There are two agents involved in the process: A Timekeeper agent computes response time values and communicates the computed response times to the record-keeping Web site. A Librarian agent stores and retrieves time samples upon request and provides the Timekeeper with an interface to perform its actions. The time samples have to be stored in cookies, a dedicated window or a frame within a browser window as the browser cannot maintain the values across page loads.

This approach allows accurate measurement of response times as experienced by the end user. There are however, three problems with this approach. Firstly, the approach does not compute response time for a request which is not made through an instrumented hyperlink. Secondly, pages containing images or PDF files cannot be instrumented. Thirdly, the approach does not work with browsers that do not support the particular scripting language used for instrumentation, or when the execution of the scripting language is disabled by the user.

## 2.2 Server-side Measurement

Cherkasova et al. (2003) proposed a tool called EtE monitor to measure Web site performance by passively collecting packet traces from a server site. The EtE monitor architecture is shown in Figure 1.

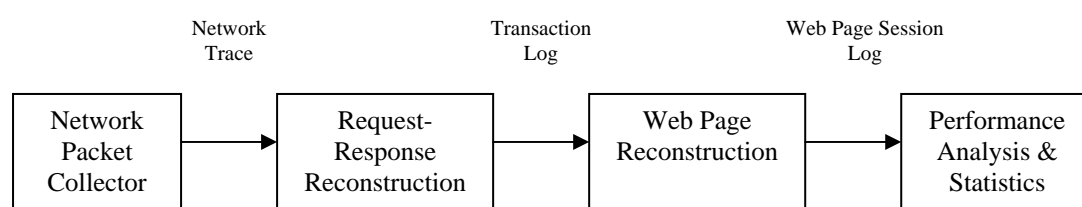


Figure 1 EtE Monitor Architecture

The network packet collector module collects network packets using tcp-dump and records them to a network trace for offline analysis. The request-response reconstruction module uses the network trace to reconstruct all TCP connections using client IP addresses, client port numbers, and request (response) TCP sequence numbers. HTTP transactions are then extracted and the HTTP header lines of each request recorded in the transaction log for future processing.

The Web page reconstruction module groups underlying physical object retrievals together into logical Web pages and stores them in the Web page session log. The process involves two passes. The first pass builds a knowledge base of Web page objects based on the chronologically sorted transaction log of successful responses (status code 200). A list of distinct client IP addresses, called Client Access Table, is compiled. For each client, the requested HTML documents and the corresponding embedded objects are identified on the basis of the request URL, request referer field, the response content type, and the client IP address to construct to Web Page Table.

The Client Access Table and the Web Page Table are then scanned to produce a knowledge base which groups accesses for the same Web pages from other clients' browsers but do not set the referer fields in the second pass of Web page reconstruction. The EtE monitor also statistically analyses the pattern of accesses in order to remove incorrectly or redundantly grouped accesses.

Finally, the performance analysis and statistics module summarises a variety of performance characteristics integrated across all client accesses. The analyses are based on critical time stamps for connection and requests.

Marshak and Levy (2003) rely on server log analysis rather than packet reconstruction. Their measurement tool consists of two elements: a measuring element and a latency estimation element. The measuring element consists of four components: a sentry, an extended server access log, a queue latency probe, and an external pinger. The sentry is a zero size inline HTTP object inserted at the end of the requested HTML documents to track the arrival time to the user. The extended server access log is used to record information about user requests and server responses. The queue latency probe estimates the latency experienced by a TCP connection waiting in a complete connection queue. The pinger is used to estimate the round trip time. The information is extracted to calculate the response time. The estimation element takes data from different components and estimates the time required for a user to fetch a Web page from the server.

In general, the server-side method could estimate response time for all the accesses to a Web site, and not be constrained by the way a Web page is accessed. The estimation is closer to the actual response time as compared to active probing. Whilst Cherkasova et. al.'s method does not require additional modification to Web pages, the method suffers from the accuracy of the heuristics used to construct the Client Access and Web Page Tables. Marshak and Levy's method, based on the HTTP level, is more economical, even though modification of the Web pages is required. Both methods have limitations in their ability to estimate response time for clients behind proxies.

### **2.3 Active Probing**

With active probing, a few geographically distributed synthetic clients (the agents) are used to periodically probe the server by requesting a set of Web pages or operations. The agents mimic users from different locations over the world. The measurements obtained are only

representations of latencies that may be experienced by the end users. Examples of active probing include Keynote (<http://www.keynote.com/>) and NetMechanic (<http://www.netmechanics.com>).

There are two major drawbacks with this method. Firstly, the method reflects only a fraction of users' experience. Secondly, the agents are actually executing on a different set of machines from the actual users accessing the tested Web site. Thus, the reliability of the results can be questioned. However, this method does provide better control over synthetic clients and is easier to use to perform measurements.

### **3. Methodology**

The method used in this study uses a combination of client and server-side measurements as well as active probing. It measures response time and decomposes the time into three of its constituents: transmission, processing/generation, and displaying/rendering. The effect of browsing context in which a Web page is accessed is examined. The following steps were followed during the study:

- A prototype Web site for an on-line bookstore was built to act as the targeting server. The server machine is a Pentium 4-2.40 GHz with 512 MB main memory, running Apache Tomcat 5.0 on Windows XP Professional SP2.
- A client Pentium 3-450 MHz with 256 MB memory machine was connected to the Web server in a LAN environment.
- Web pages of the on-line bookstore were instrumented using JavaScript to record the time taken to load (display) the pages, denoted as  $T_d$ , measured by an onLoad event handler triggered when the body of the page is loaded completely. (Code given in Figure 2.)

- A Muffin (<http://muffin.doit.org>) proxy server was configured to record the time elapsed between the request for an object and the receipt of the object by the client, denoted as  $T_s$ . Muffin is a WWW filtering system written in Java.
- The server access log was extended to record additional information to facilitate measurement of server processing time for page generation, denoted as  $T_p$ .
- Two Web pages (home page and searching) were accessed from the client machine. The corresponding time instances were recorded.  $T_d$ ,  $T_s$ , and  $T_p$  were then computed based on these times and analysed. Three sets of data were collected for each test and the average was used for the computation and analysis.
- The measurement was repeated with different browsing contexts. The contexts studied in this paper include the number of browser windows opened concurrently, how they are used, and whether a requested page or object has previously been cached.

```

var start = new Date().getTime();
var finish;

function loadtime() {
    finish = new Date().getTime();
    return finish - start;
}

function tester() {
    var ok= confirm ("Milliseconds since page started
                    loading at " + start + ": " + loadtime() +
                    ", finished loading at " + finish);
    if (!ok)
        tester();
}

```

Figure 2 JavaScript Code to Measure Page Load Time

The experiment setup is illustrated in Figure 3.

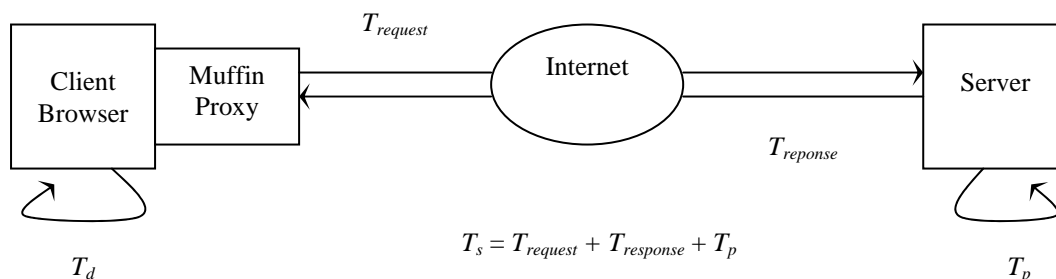
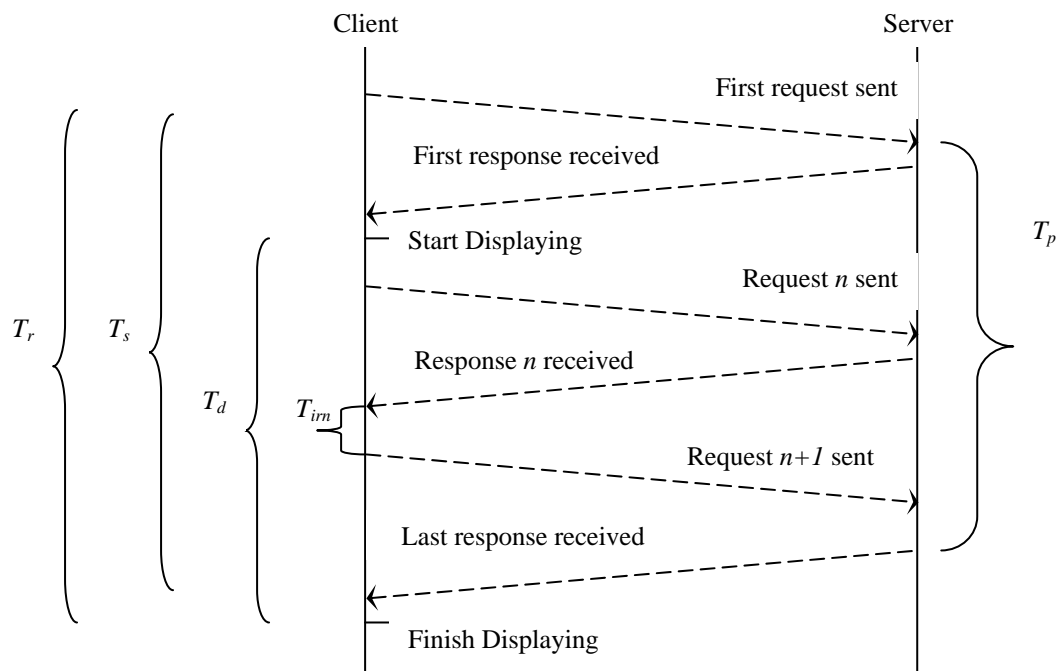


Figure 3 Experiment Setup

It has to be noted that the measured times could overlap. For example, an object could be requested while others are being served by the server and while the client is displaying yet other objects. Therefore, two other notions are introduced here:

- $T_{ir}$ , the time between successive requests when no previous request is being processed or served.
- $T_{es}$ , the effective (actual) time taken to serve all the requested objects.

Another important issue was that because Java Server Pages (JSP) was used to develop the prototype Web site and the first access to a JSP page requires compilation of the page into Java source code and then Java Servlet, the first access evidenced a longer response time. This effect of “first access” was studied. Time instances measured are indicated in Figure 4.



Effective serve time,  $T_{es} = T_s - \sum_{n=1}^k T_{ir_n}$ ,  $k$  is the number of objects requested

Figure 4 Time Instances Measured

#### 4. Data Analysis

Below is a summary of notations used:

$T_r$  : Response time, the time elapsed from the first user request is sent until the last object returned is displayed.

$T_{es}$  : Effective serve time, the actual time taken to download (serve) all the requested objects.

$T_{ir}$  : Inter-request time, the “idle” time between responses received and the subsequent requests sent.

$T_s$  : Serve time,  $T_{es} + T_{ir}$

$T_p$  : Processing time, the time taken by the server to process a user request until the corresponding response is returned

$T_d$  : Display time, the time taken by the Web browser to display a Web page.

##### 4.1 Home Page

Table 1 below shows data collected for the accesses to the home page. An identical HTML version of the home page was developed to compare its performance with its JSP counterpart.

Table 1 Times for the Accesses to the Home Pages in Milliseconds

<b>Time</b>	<b>First Access to the JSP Home Page</b>	<b>Subsequent Access to the JSP Home Page without Caching</b>	<b>Subsequent Access to the JSP Home Page with Caching</b>	<b>Access to the HTML Home Page without Caching</b>	<b>Access to the HTML Home Page with Caching</b>
$T_r$	6983	584	481	387	260
$T_{es}$	2847	167	157	160	97
$T_{ir}$	4043	350	281	203	153
$T_s$	6890	517	438	363	250
$T_p$	6029	214	120	10	0
$T_d$	4093	528	427	307	234

Taking  $T_r$  for accessing the HTML page with caching as a reference value, the ratios of  $T_r$ 's to this reference value (260), across the first row of the table, from left to right (6983, 584, 481, 387, 260), are 26.9, 2.2, 1.9, 1.5, and 1.0 respectively. The need to compile and interpret the JSP page for the first access caused the longest  $T_r$  measured, about 7 seconds. Fortunately, this is only experienced by the first access after the server has started. Furthermore, if the server was shut down (without physically shutting down the server machine) but the compiled Java Servlets were not cleared from the work directory,  $T_r$  for the first access to the JSP home page will be reduced to about 1300 ms. It is therefore wise to make dummy

accesses to the JSP pages to force the JSP container performs compilation and interpretation in advance before actual users access the pages.

Of more concern is the effect of caching and page dynamism on  $T_r$ . Accessing a static page without caching is 1.5 (584/387) times faster than accessing its dynamic counter part. With caching, the improvement is 1.9 (481/260). It is interesting that  $T_r$  for a non-cached static page (387) is 1.2 times faster than a dynamic page with caching (481).  $T_p$ ,  $T_d$ , and  $T_{ir}$  seem to be the major contributing factors to this phenomenon, with the ratios (cached dynamic to non-cached static) of 12 for  $T_p$  (120/10) and 1.4 for  $T_d$  (427/307) and  $T_{ir}$  (281/203). We could deduce that the need to process a JSP page before the relevant cached objects are retrieved is the major factor causing the longer response time. However, the phenomenon may change the other way if transmission over the Internet is involved as  $T_{es}$  will increase for non-cached accesses. The corresponding  $T_{es}$  are about the same for both cases in our study.

For a dynamic page, caching could improve  $T_r$  by a factor of 1.2 in our experiment (584/481). For a static page, the improvement is by 1.5 (387/160). Caching shows a better  $T_r$  improvement for the access to a static page as compared to a dynamic page because  $T_{es}$  is a more significant contributor to  $T_r$  in the downloading of a static page than the downloading of a dynamic page, and the effect of caching is mainly on reducing  $T_{es}$ . This is demonstrated in the ratio of  $T_{es}$  to  $T_r$  for the dynamic page with caching is 0.33 (157/481), an increment from 0.29 (167/584) for the dynamic page without caching; while for the cached static page, the ratio is 0.37 (97/260), an improvement from 0.41 (160/387) for the static page without caching. Increment of the ratio for downloading a dynamic page is due to a more significant reduction in  $T_p$  (43.9%) than  $T_{es}$  (6.0%). In the downloading of a static page, the reduction in  $T_{es}$  is 39.4%, a much higher reduction than that is for a dynamic page. Since  $T_p$  contributes

little to  $T_r$  in the downloading of a static page (3.6%), even though caching improves  $T_p$  by 100% in this case, it is not significant enough to reflect the improvement in  $T_r$ . Nevertheless, the fact is that caching reduces actual time for  $T_p$  and  $T_{es}$  for both dynamic and static pages.

To compare  $T_d$  among different Web page accesses, the ratios of  $T_d$  to  $T_s$ ,  $T_d$  to  $T_{es}$ , and  $T_d$  to  $T_p$  are used instead of  $T_d$  itself. The reason is that  $T_d$  may overlap with or even due to  $T_s$ , which consists of  $T_{es}$ ,  $T_p$ , and  $T_{ir}$ . Higher ratios imply higher significance for  $T_d$ . Table 2 shows the ratios.

Table 2 Ratios of  $T_d$  to  $T_s$ ,  $T_{es}$  and  $T_p$

$T_d$ To	First Access to the JSP Home Page	Subsequent Access to the JSP Home Page without Caching	Subsequent Access to the JSP Home Page with Caching	Access to the HTML Home Page without Caching	Access to the HTML Home Page with Caching
$T_s$	0.59	1.02	0.97	0.85	0.94
$T_{es}$	1.44	3.16	2.72	1.92	2.41
$T_p$	0.68	2.47	3.56	30.7	-

As can be seen from Table 2, the ratios show that  $T_d$  is more significant for dynamic pages, except for the first access to the JSP page as a large amount of time is occupied for server processing. It is therefore worth studying ways to enhance efficiency of displaying for dynamic pages since Web sites have more emphasis on interactivity and dynamism nowadays.

## 4.2 Search Page

Table 3 shows data collected for performing database search. Comparisons are made between searches with different resulting response sizes and searches with different number of keywords.

Table 3 Times for Database Search in Milliseconds

<b>Time</b>	<b>Searching with 1 Keyword, Response Size &lt; 100 kB (Test 1)</b>	<b>Searching with 1 Keyword, Response Size &gt; 150 kB (Test 2)</b>	<b>Searching with 3 Keywords, Response Size &lt; 30 kB (Test 3)</b>	<b>Searching with 3 Keywords, Response Size &gt; 120 kB (Test 4)</b>
$T_r$	2056	2324	1639	2006
$T_{es}$	1525	1656	180	1442
$T_{ir}$	418	614	1419	451
$T_s$	1943	2270	1599	1893
$T_p$	1477	1490	1391	1380
$T_d$	451	715	143	491

A first glance at the result shown in Table 3 suggests that the resulting response size determines the response time. However, more detailed investigation shows that the number of objects returned is more important. In one of the data samples, an object of size 18 kB took 40 ms to be served while another object of size 8 kB took 270 ms. Similar results were found throughout other data samples. Differences in the times measured are largely due to the number of objects returned for the particular request. For the four tests shown in Table 3, the numbers of objects returned are 14, 21, 4 and 14 respectively. Test 1 and Test 4 (both with 14 objects returned) have very similar results. Test 2 (21 objects returned) takes the longest time among the tests while Test 3 (4 objects returned) takes the shortest time.

If  $T_{ir}$  for the four tests are examined, it is evident that the more objects are returned, the longer the  $T_{ir}$  will be. This is because there is a time gap between some responses received and subsequent requests sent. Even though some requests may be made while the client is waiting for previous responses, an increment in the number of objects requested will inevitably increase the total  $T_{ir}$  for that Web page access. The ratio of  $T_{ir}$  to  $T_r$  for Test 2 (with the most objects returned) is 0.26 as compared to 0.11 for Test 3 (with the least objects returned). For Test 1, the ratio is 0.20 and for Test 4, it is 0.22. However, we have yet to study the impact of a single extremely large object as compared to many smaller objects that amounted to the same eventual size.

### 4.3 Browsing Context

We have examined two browsing contexts: the number of Web browser windows opened concurrently and how the Web browsers are used. To study the effect of the number of Web browser windows opened concurrently, we accessed the JSP and HTML home pages with both five and ten Web browser windows opened. Table 4 shows the results for accessing JSP and HTML home pages with five and ten browser windows opened concurrently.

Table 4 Accessing Web Page with Multiple Web Browser Windows Opened Concurrently

<b>Time</b>	<b>JSP: 5 Browser Windows</b>	<b>HTML: 5 Browser Windows</b>	<b>JSP: 10 Browser Windows</b>	<b>HTML: 10 Browser Windows</b>
$T_r$	594	410	591	410
$T_{es}$	94	50	80	80
$T_p$	187	5	188	0
$T_s$	507	350	491	360
$T_d$	417	370	441	370

From the results, there is no significant difference between  $T_r$  for Web page accesses when different numbers of Web browser windows are opened concurrently.

To study the effect of how the Web browsers are used, we opened five Web browser windows and accessed the JSP home page. We examined the effect of whether a browser window had previously accessed to the page and the number of intermediate accesses to other Web sites/pages. The following four tests were used:

- Test 1: Immediate access to the page using the same browser window.
- Test 2: Immediate access to the page using the same browser window by clicking refresh button.
- Test 3: Immediate access to the page using another browser window that has already opened with the current one remained opened.
- Test 4: Immediate access to the page using another browser window that has already opened with the current one closed.

- Test 5: Access to the page using the same browser window with five intermediate accesses to other Web sites/pages.
- Test 6: Access to the page using the same browser window with ten intermediate accesses to other Web sites/pages

It has to be pointed out that for Test 1, as the Web page is retrieved from local cache, the request will neither go through the Muffin filter nor be processed by the server. Thus, only  $T_d$  could be measured.  $T_r$  is assumed to be the same as  $T_d$  in this test. Table 5 shows the results.

Table 5 Different Context in which a Web Page is Accessed

<b>Time</b>	<b>Test 1</b>	<b>Test 2</b>	<b>Test 3</b>	<b>Test 4</b>	<b>Test 5</b>	<b>Test 6</b>
$T_r$	60	451	230	621	230	271
$T_{es}$	0	90	10	100	20	10
$T_p$	0	0	0	172	0	0
$T_s$	0	401	10	591	20	10
$T_d$	60	421	110	431	90	130

It can be seen from Table 5 that once a Web browser window is closed, subsequent access to the page it contained will take longer time to complete its downloading. There are two reasons for this observed phenomenon. Firstly, the server needs to re-process the dynamic JSP page. Secondly, the client browser needs more initial time before rendering the page. This is proven by the longer time it takes for the browser to start displaying the page after the first response is received. Therefore,  $T_r$  for Test 4 (requires server re-processing and more initial browser rendering time) is larger than Test 2 and Test 3.

The number of intermediate activities before a page is reloaded in the same browser window has more significant impact on  $T_d$  than other times. However, there are a few factors that may affect the result:

- The size of the cache and the caching policy/algorithm used.

- The characteristics of the intermediate activities such as how extensively the system resources are used and the time taken by the intermediate activities.

## 5. Conclusions and Future Work

Most existing studies use active probing, client-side or server-side methods for the measurement of Web-based system performance. This study combines the three methods to disassemble system response time into its constituents: *transmission*, *processing/generation*, and *displaying/rendering*. With this method, the problem experienced by a Web site in terms of response can be better understood and the cause of the problem can be identified more easily. Based on the measurement, appropriate remedial actions can be chosen to relieve the problem. Another aspect of this study was an investigation into the relationship between browsing context and Web page performance in a quantitative manner.

This study also shows a few aspects pertinent to dynamism of Web pages or interactivity of Web sites. Caching could shorten response time for dynamic Web pages and thus gives better user browsing experience. Dynamic Web pages not only take longer for the requests to be processed at the server side, but also take longer for the pages to be displayed at the client side. The number of objects returned from a request does affect response time and the impact is larger than the total size of objects returned. Therefore, reducing unnecessary embedded objects in a Web page could enhance response time. Meanwhile, the number of browser windows opened concurrently does not have significant impact on response time, but how these browser windows are used does.

This study has a few limitations. Firstly, the client machine had to be instrumented with a Muffin filter to records times. Secondly, measurements of different response time

components were not automated and integrated. Thirdly, the experiment was conducted in a LAN environment where it is hard to assess the effect of network conditions on response time.

Further work for this study would be to investigate ways to integrate different components of the measurement tool as an automated software agent that is able to receive and extract necessary information from both client and server sides. The agent will perform calculations and present the customisable results in a comprehensive manner. The study will be extended for the measurement of response time for real Web sites. Currently, such a tool is being designed by using HTML frames and JavaScript.

## References

- Cherkasova, L., Fu, Y., Tang, W., and Vahdat, A. (2003) Measuring and Characterizing End-to-End Internet Service Performance. *ACM Transactions on Internet Technology* Vol. 3, No. 4, pp 347-391.
- Debaraj, S., Fan, M., and Kohli, R. (2003) E-Loyalty - Elusive Ideal or Competitive Edge? *Communications of the ACM* Vol. 46, No. 9, pp 184-191.
- Dyson, P., and Longshaw, A. (2004) *Architecting Enterprise Solutions: Patterns for High-Capability Internet-based Systems*. Wiley, Chichester.
- Lilja, D.J. (2000) *Measuring Computer Performance: A Practitioner's Guide*. Cambridge University Press, Cambridge.
- Marshak, M., and Levy, H. (2003) Evaluating Web User Perceived Latency Using Server Side Measurements. *Computer Communications* Vol. 26, No.8, pp 872-887.
- Nielsen, J. (2000) *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing, Indianapolis.
- Rajamony, R., and Elnozahy, M. (2001) Measuring Client-Perceived Response Times on the WWW. In *3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, San Francisco, California.