AT&T Labs - Research

subject: **Analyzing factors that influence end-to-end Web performance**
Work Project No. 539054-0000

date: **November 29, 1999**

from: **Balachander Krishnamurthy**
Dept. HA1630000
FP D229
973-360-8648
bala@research.att.com
HA1630000-991129-07TM

Craig E. Wills
Worcester Polytechnic Institute
Worcester MA
cew@cs.wpi.edu

TECHNICAL MEMORANDUM

With the growing popularity of the Web, performance issues have come to the fore both on an intranet basis and on an Internet basis. Web performance impacts the popularity of a particular Web site or service as well as the load on the network. While Web caching has become popular and has been studied extensively, there have been no publically available *end-to-end* measurements that have focused on a large number of popular Web servers examining the components of delay or the effectiveness of the recent changes to the HTTP protocol. In this paper we report on an extensive study carried out from several client sites geographically distributed around the world to a collection of over 700 servers to which a majority of Web traffic is directed. Our results show that the primary changes in the recently draft standardized HTTP/1.1 protocol are indeed useful but further examination is warranted.

1 Introduction

Several projects have studied factors that influence the performance of the Web using Web server logs, proxy logs and client traces. Over the last four and a half years, significant effort has gone into improving the protocol on which the Web is based and the new version of HTTP, namely HTTP/1.1 has been recently draft-standardized [1] . However, to the best of our knowledge, no one has performed an end-to-end test of the Web in terms of the factors that influence performance as perceived by users when they visit various Web sites.

This work grows out of independent work by the authors for testing different aspects of Web performance. It is a natural followup work to the recently concluded PROCOW study [2] , a large scale study examining if the Web servers running in popular Web sites around the world claiming to run HTTP/1.1 were indeed compliant with the HTTP/1.1 protocol specification. The PROCOW study examined the compliancy of the servers by sending valid HTTP/1.1 client requests from several places in the world. It also follows up on the methodology of the work examining content reuse and server responses relevant for Web caching [3] .

In this work we use the PROCOW infrastructure of eight client sites around the world (and additional clients in more countries) to test how various factors affect the *performance* of the Web. Our study is centered on the key set of changes between HTTP/1.0 and HTTP/1.1 [4] to test their impact on performance. We also examine their impact on performance in conjunction with other factors such as caching and distributed Web content.

Our study is significant in that it examines the performance of various protocol options by measuring end-to-end response for actual Web servers from a variety of client sites. This is a significantly broader study compared to that of Nielsen, et al [5] , which measured the performance of HTTP/1.0 and HTTP/1.1 in a controlled setting for a single synthesized Web page. Our study also examines the contribution of other factors to end-to-end performance, such as caching, multi-server content and byte range requests.

We gathered a set of active measurements from the various client sites around the world to over 700 popular sites to help quantify the benefits of HTTP/1.1 features in conjunction with other factors. These include persistent connections, persistent connections with pipelining, range requests, caching and responses from multiple servers for a single Web page.

The collection of Web servers tested are representative of the *interesting* set of Web servers since we attempted to include servers to which a significant portion of the request traffic is addressed. We do not claim our client sites are representative of all users, but they do include a variety of sites. We ensured that there is some degree of control in our experiment but given that the Web experience changes fairly often, our study mimics real life. In analyzing the results, we look for trends rather than absolute numbers. The fact that the results are relatively consistent over multiple clients and that we make some measurements from the same client/server pair over multiple days at the same time gives us some degree of belief in the repeatability of our experiments.

In the remainder of this paper we describe the factors studied in our work, followed by a discussion of the methodology we used in performing the study. The middle portion of the paper presents the results from our study on the test sets we use followed by a discussion on possible implications of these results for Web servers, caches and the HTTP protocol. The paper concludes with a description of related and future work.

2 Study

There are numerous issues involved in the end-to-end performance of retrieving a Web page over the Internet. In the following we describe the specific issues we study (and those we do not) in this work and our reasons for doing so. In addition there are many issues that we do not explicitly study, but consider in the testing and analysis of the various factors.

2.1 Factors Studied

The issues we explicitly study are:

- *Protocol Options.* What is the effect of different HTTP options on performance. HTTP/1.0 has been used by clients and servers for a long time, but clients and servers have been moving to HTTP/1.1. The de facto standard use of HTTP/1.0 has been for clients to open multiple parallel connections to the same server in order to improve performance over serialized connections. In HTTP/1.1 persistent connections are supported, which can either be used for serialized request/response pairs or for pipelining multiple requests and then receiving multiple responses. The relative performance of these protocol options was studied by Nielsen, but it was done on a limited basis with a single artificially created page with 40 embedded objects [5] . The experiment was useful to isolate specific low level issues on the impact of TCP on HTTP/1.1 and demonstrated the usefulness of pipelining/persistent connections under the test conditions. A primary goal of our work is to study these protocol options using real-life servers.
- *Caching.* We do not study expected cache hit rates nor particular policies for cache replacement and coherency. These issues have been studied in numerous pieces of previous work such as [6, 7] . Rather we focus on studying the relative performance gains for different levels of caching effectiveness. These relative gains may vary according to the protocol option being used, the cache hit rate and whether validation of cached contents is required.
- *Multi-server content.* Rather than serve the base page and all embedded images from the same base server, Web sites are serving some or all of the embedded objects from different servers—both servers located at the same site and on servers located at different sites. This approach reduces the load on the base server, but does it improve the response time for the end user?
- *Byte range requests*

  One of the improvements in HTTP/1.1 was the introduction of `Range` requests, whereby only a part of a resource can be downloaded instead of the entire resource. Obviously, there is a clear gain in bandwidth usage since fewer bytes are shipped. But does this translate into significant savings in user perceived latency? For example, would a proxy cache be better off retrieving the entire resource in response to a client request for a specific byte range request? If the time for a byte range request is not significantly faster then the proxy could retrieve the entire contents for possible later use.

2.2 Factors Considered

In studying these issues there are numerous other factors that must be accounted for in our testing and analysis. These factors are:

- *Network delay.* The magnitude and variance of network delay between a client and server can have a major influence on the total response time for serving a set of Web objects from the server to the client. In an ideal study we would be able to control this variable as we conduct various tests, but to do so requires an artificial testbed, which would raise questions about the applicability of any results to actual networks. Rather, our approach is to test with real networks and account for effects of variable network delays in our analysis. We test different protocol options to the same server one after another to ensure each is tested under approximately the same network conditions. We also tested client/server pairs at different times of day in an attempt to capture some variance in network conditions.

- *Server load.* The server load can have a major influence on the time to service a request. As is the case for network delay, we do not control, or even know, the load of a server under test, but we always test different protocol options between a client and a server at approximately the same time and we do vary the time of day in which testing is done.
- *Number of objects and total bytes.* The time to retrieve a Web page is dependent on the number and size of objects it contains. In our study we do not specifically control this variable, but rather we select a test set of Web pages and then account for the number and size of objects in our analysis.

2.3 Factors Not Examined

There are many other issues influencing the end-to-end performance in retrieving a Web page. The following are some that we considered for our study, but did not include in our study at this time for the reasons indicated.

- *DNS lookup time.* Once a server to IP address mapping has been obtained and is cached, the DNS lookup time is a trivial contribution to the overall performance. If the mapping is not cached then this contributes some time to overall performance. Because we expect most mappings to be cached in our study we did not explicitly study the lookup time.
- *Redirection.* Some Web sites use HTTP redirection to direct user requests to another URL. We did not consider the cost of this redirection, but rather dropped such sites from our study.
- *Dynamic content.* The amount of time for a server to serve an object stored as a static file should be less than the time to serve an object of the same size that must be dynamically generated. The impact of dynamic content on end-to-end performance would be nice to study, but has not been done as yet.
- *Packet level performance.* Examining packet level performance through tools such as *tcpdump* would be useful in understanding reasons for observed delays and variance in network performance. However, to keep a simpler test set up for distribution to a number of client sites around the world we chose not to include packet level performance issues at this time.

3 Methodology

Our basic methodology is to make active measurements on the effect of different protocol options for a large number of client/server pairs at different times. The previous study on HTTP/1.1 performance improvements used an artificial environment for testing [5] . Other Web performance studies have used logs and packet traces to obtain timing information, but this approach does not allow a controlled set of requests to be issued. Our approach was to identify a set of client and server sites for our tests.

We came to an early conclusion that it was virtually impossible to have a set of representative client sites since information about the distribution of clients and their network connections is both hard to obtain and verify. Additionally, to obtain a fair sampling of clients around the world would involve significant effort. We chose to sample from a set of client sites where we had professional connections. The client sites used in our study and their location are

1. AT&T Research Labs, NJ USA, with multiple T-1 connections to the Internet.
2. A commercial site in Santiago, Chile (10 Mbps via fiber to Telefonica Net which has a slower link to the Internet and links to C&W (Cable and Wireless) and Alternet via two hops of 45 Mbps ATM link
3. Hewlett-Packard Labs, Palo Alto, CA, USA
4. ACIRI: AT&T Center for Internet Research at ICSI, Berkeley, CA, USA (10 Mbps link from ACIRI to UCB, then UCB to Internet over Calren)
5. University of Kentucky, Lexington, Kentucky, USA (connected via UUNET over a DS3 at 45 Mbps link)
6. A site belonging to a academic network UNINETT AS, in Trondheim, Norway (10 Mbps link to UNINETT-GW and increasing speed links (4) to the Internet via NORDUnet)
7. University of Western Australia, Nedlands, Western Australia
8. A private site in Cape Town, South Africa (64K Digital connection similar to US-standard 56K connection, to the Internet)
9. Worcester Polytechnic Institute, Worcester, MA with multiple T-1 links to the Internet.

The choice of servers is a bit easier since there seems to be necessary concurrence in the notion of "popular" sites. Advertisers depend on this information and sites would attempt to demonstrate that studies significantly lowering their standing in any ranking are incorrect. Accordingly, we went with a subset of the collection done for the PROCOW study [2] , which used a combination of collection techniques. Briefly, it merged a combination of recognized rating sites (MediaMetrix [8] , Netcraft [9] , Hot100 [10]) and a set of sites that are likely to be popular given their business prominence (Fortune 500 [11] and Global 500 [12]). The end result was a list of 711 popular server sites for which we retrieved the home page and all embedded objects. In creating the list, we did not try to distinguish whether a server site was supporting HTTP/1.0 or HTTP/1.1.

The basic engine for making all retrievals in our study is httperf [13] . We obtained a publicly available copy of the software and modified it slightly to print out additional information needed for our study. The httperf software is attractive because it allows a set of objects to be retrieved from a server using the variety of 1.0 and 1.1 protocol options of interest to our study. The native software collects and prints out a number of statistics about the status and performance of retrieving the set of objects. Of particular interest to our study is that it records the number of server connections made and for each retrieved object, the time the request was sent, the time receipt of the response began and the time the complete response was received. For small objects the last two times may be the same.

The following algorithm describes the method used for a single test between a client and a server. It exploits features of httperf and overcomes certain limitations of httperf for the purposes of our study. The two primary limitations that we had to overcome was that httperf does not parse HTML code to retrieve embedded images and a single run of httperf could communicate with only one server. However exploited httperf's feature that it can retrieve a fixed set of URLs from a server using serial requests over separate connections, parallel requests over separate connections, serial requests over a persistent connections and pipelined requests over a persistent connections. The resulting algorithm starting with a base URL for a given server is as follows:

```
1. Use httperf to retrieve the base URL from the server and store the results.
2. Parse the base URL code to determine all unique embedded objects.
3. Separate the base and embedded objects according to their server.
4. For each server containing needed objects {
    5. (serial-1.0) Use httperf to retrieve all objects using
       serialized HTTP/1.0 requests.
    6. (burst-1.0) Use httperf to retrieve all objects using
       up to four parallel HTTP/1.0 requests.
    7. (serial-1.1) Use httperf to retrieve all objects using
       serialized requests over an HTTP/1.1 persistent connection.
    8. (burst-1.1) Use httperf to retrieve all objects using
       pipelined requests over an HTTP/1.1 persistent connection.
}
```

There are a number points to note about the algorithm. The initial retrieval in Step 1 is used to determine the set of objects to fetch. If all embedded objects are from the same server as the base URL then there will be only one server list in Step 3. The server list for the base server includes the base object. The burstiness of parallel connections in Step 6 and pipelined connections in Step 8 does not begin until the first object in the list is retrieved. Step 1 retrieves and stores the object contents. Steps 5-8 retrieve, but do not store object contents. Steps 5-8 are used for all performance measurements.

We used four parallel connections in the burst-1.0 method since that appears to be the default in the popular browsers (Netscape and Internet Explorer). With each additional parallel connection, there is a necessary load on the client and the server (more on the server since it has to have free TCP slots to handle several client connections in parallel).

This basic test is used to compare performance of each protocol option for a specific client and server. While the comparison of response times for an individual test may not be meaningful due to variation in network and server load, we use these individual tests as building blocks for measuring the relative performance of the protocol options over a large number of tests.

We used this basic test over two sets of test data. All tests were made in November, 1999. The first set of test data consists of the 711 previously identified servers. Tests of each of these servers were run once from each of the client sites. The tests last several hours so each client/server test may be run under different network conditions, but the four protocol options of each test are run under approximately the same conditions.

In addition, we ran the test in a more controlled setting from the AT&T, Chile and WPI client sites. For these tests we selected 72 server sites from a list of 200 sites identified on the current MediaMetrix, Netcraft and Hot100 sites. The selected sites were chosen because they supported pipelining and persistent connections in a test run on these sites. The controlled tests were run on the same fixed 6 hour intervals from each site for a week. This controlled test was designed to study caching and time of day factors.

4 Results

This section provides results to address the issues for study that were identified in Section 2.

4.1 Test Sets

The initial part of our results is analyzing the base set of statistics for our client/server test sets, and is shown in Table 1 for client/server pairs. The clients are our test sites, the servers are either PROCOW indicating we tested the 711 servers from the earlier PROCOW study or "select" indicating we tested with 72 servers repeated every six hours.

Table 1: Test Sets

Client/ Server Set	200 Status Base URL	200 Status Servers	Multiple Object Servers	Some Persistence Servers (%)	Perfect Persistence Servers (%)
att/procow	670	855	674	288 (43%)	167 (25%)
aciri/procow	673	858	667	296 (44%)	223 (33%)
aust/procow	667	854	664	257 (39%)	201 (30%)
chile/procow	674	862	671	274 (41%)	200 (30%)
hp/procow	665	854	662	274 (41%)	201 (30%)
uky/procow	645	824	635	280 (44%)	196 (31%)
norway/procow	668	856	662	166 (25%)	128 (19%)
safrica/procow	663	848	654	286 (44%)	194 (30%)
wpi/procow	657	834	662	258 (39%)	192 (29%)
att/select	1515	2588	1975	1064 (54%)	858 (43%)
chile/select	1873	3161	2423	1198 (49%)	910 (38%)
wpi/select	1897	3223	2456	1323 (54%)	1049 (43%)

Focusing on the PROCOW test set, the second column indicates the number of servers out of the 711 that returned an HTTP 200 response code for the initial retrieval. Those not returning this value either returned 302 (redirection), 404 (not found), or the client timed out. Once a base object is retrieved, all objects from one or more servers are retrieved. A count of these servers returning a 200 response code for all objects is shown in column 3 of Table 1. The fourth column shows the number of these servers that return more than one object. We focus on these servers because persistent connections are not meaningful if only one object is retrieved. The last two columns in Table 1 show the number and percentage of multiple object servers exhibiting persistent connection.

The last two columns need further explanation: one focus of our study is to compare the performance of the four protocol options. Thus for a client/server test of n objects, we only want to consider cases when the number of retrieved objects r is equal to n for all protocol options. If one or more of the four protocol option tests retrieves fewer than n objects (i.e., $r<n$) then we discount that test (all four protocol options) for further study. In many of these failed cases, the server does not support persistent connection or handle pipelined requests, but in some cases one or both of the HTTP/1.0 options fail. In addition to the number of objects retrieved we also examine the number of TCP connections c that are used. For both HTTP/1.0 options $c{=}n$, but if $r{=}n$ and $c{=}n$ for the HTTP/1.1 options then all objects have been retrieved, but not with any persistent connections. These test cases are also eliminated from further study. The remaining tests are classified as showing "some" persistence. These tests yield $r{=}n$ for all protocol options and $1 \leq c$ for both 1.1 options indicating that fewer connections than objects were needed. Of this category, we classify tests with $c{=}1$ for both HTTP/1.1 tests as "perfect" indicating all objects are retrieved in a single persistent connection. We use the test results showing some and perfect persistence in making comparisons between the four protocol options in our work.

The last three rows of Table 1 show the same statistics for the selected set of servers tested periodically from three client sites. The last two columns indicate that these tests are a bit better in exhibiting a higher percentage of persistence, but the percentages are not as high as we would expect from a "select" group. These reduced numbers come from the inclusion of servers not exhibiting persistence in our initial tests due to an early error (subsequently fixed) in one of our analysis scripts. In addition, just because the base server supports persistence, it is possible that other servers providing its embedded objects may not. Again only tests exhibiting persistence are considered for further analysis.

As a final point on summarizing the test sets we note that the Nielsen et al., paper on HTTP/1.1 also described two other performance improvements—cascading style sheets (CSS) and portable network graphics (PNG). As an interesting sidelight to our study we examined penetration of these improvements to the set of PROCOW servers. Examination of the AT&T results (typical for other results) showed 82 (12%) of the 670 base URLs using style sheets. A similar examination for PNG images found zero usage.

4.2 Protocol Options

In examining the various issues influencing end-to-end Web performance we first examined the impact of the four protocol options. For this analysis we only consider the client/server tests exhibiting persistence. Results for the four protocol options from three of the client sites using the PROCOW test set are shown in Tables 2 and 3. Table 2 is restricted to only servers that exhibit perfect persistence in the retrieval of objects. Table 3 includes all servers exhibiting some persistence as previously defined. Results from the remaining client sites are shown in Tables 7 and 8 at the end of the paper.

Table 2: Servers Exhibiting Perfect Persistence from Three Client Sites

Client Site	Object Count Range	Pct. Persistent	Ave. Obj. Cnt.	Ave. Obj. Bytes	Ave. Retrieval Time (Sec.)(Rate with Burst-1.0)			
					Serial-1.0	Burst-1.0	Serial-1.1	Burst-1.1
att	2-5	47%	3.2	18033	1.83 (1.1)	1.65 (1.0)	1.68 (1.0)	1.73 (1.0)
att	6-15	25%	9.4	50357	2.96 (1.3)	2.28 (1.0)	1.74 (0.8)	1.40 (0.6)
att	16+	03%	16.6	89834	3.71 (1.4)	2.70 (1.0)	1.72 (0.6)	1.21 (0.4)
att	Total	25%	5.7	30721	2.25 (1.2)	1.88 (1.0)	1.70 (0.9)	1.61 (0.9)
chile	2-5	47%	3.2	17082	9.23 (1.5)	6.27 (1.0)	6.45 (1.0)	6.24 (1.0)
chile	6-15	24%	9.4	49761	23.45 (2.0)	11.73 (1.0)	12.08 (1.0)	9.13 (0.8)
chile	16+	18%	24.6	79467	45.11 (2.9)	15.52 (1.0)	23.51 (1.5)	13.06 (0.8)
chile	Total	30%	9.4	38486	20.47 (2.1)	9.59 (1.0)	11.53 (1.2)	8.42 (0.9)
wpi	2-5	44%	3.2	16257	5.24 (1.1)	4.90 (1.0)	3.69 (0.8)	3.60 (0.7)
wpi	6-15	24%	9.6	52032	14.23 (1.7)	8.15 (1.0)	7.49 (0.9)	5.87 (0.7)
wpi	16+	18%	24.8	80423	26.87 (2.1)	12.87 (1.0)	14.28 (1.1)	8.20 (0.6)
wpi	Total	29%	9.6	39198	12.24 (1.6)	7.46 (1.0)	6.97 (0.9)	5.18 (0.7)

Focusing on Table 2, there are four lines of results for each client site. For each client, the first three lines are classifications based on the number of objects to be retrieved while the fourth line is a summary for all tests exhibiting perfect persistence. The categorization is introduced to examine variations that occur due to the number of objects retrieved. The ranges of 2-5, 6-15 and 16+ are intended to reflect a small, medium and large number of objects for retrieval. The third column in the table indicates the percentage of servers exhibiting perfect persistence among all servers with the given range of objects. For example, 47% of servers with 2-5 objects showed perfect persistence when tested from the AT&T client. The fourth and fifth columns show the average number of objects and bytes served by servers for the given range of objects. The last four columns show the average retrieval time for each of the four protocol options in seconds. The number in parentheses is the ratio of the given time to the time for the burst-1.0 option. The ratio is intended to show the relative performance of each option relative to common HTTP/1.0 usage. Again illustrating with an

Table 3: Servers Exhibiting Some Persistence from Three Client Sites

Client Site	Object Count Range	Pct. Per-sistent	Ave. Obj. Cnt.	Ave. Obj. Bytes	Ave. Retrieval Time (Sec.)(Rate with Burst-1.0)			
					Serial-1.0	Burst-1.0	Serial-1.1	Burst-1.1
att	2-5	53%	3.3	18701	2.40 (1.4)	1.69 (1.0)	1.78 (1.1)	2.04 (1.2)
att	6-15	40%	9.9	48740	3.64 (1.4)	2.60 (1.0)	2.22 (0.9)	1.85 (0.7)
att	16+	35%	25.1	82891	8.45 (1.9)	4.36 (1.0)	4.15 (1.0)	2.38 (0.5)
att	Total	43%	11.7	46213	4.55 (1.7)	2.74 (1.0)	2.61 (1.0)	2.09 (0.8)
chile	2-5	54%	3.3	18118	10.45 (1.6)	6.61 (1.0)	7.08 (1.1)	7.09 (1.1)
chile	6-15	39%	10.0	50172	26.70 (2.0)	13.46 (1.0)	15.14 (1.1)	13.89 (1.0)
chile	16+	30%	25.2	79829	51.10 (2.7)	18.65 (1.0)	27.33 (1.5)	21.21 (1.1)
chile	Total	41%	10.9	43108	25.58 (2.2)	11.65 (1.0)	14.61 (1.3)	12.66 (1.1)
wpi	2-5	51%	3.3	17805	5.31 (1.1)	4.82 (1.0)	4.15 (0.9)	3.96 (0.8)
wpi	6-15	38%	10.0	50788	15.96 (1.9)	8.46 (1.0)	8.97 (1.1)	8.26 (1.0)
wpi	16+	29%	25.4	79179	29.22 (2.3)	12.78 (1.0)	15.69 (1.2)	12.26 (1.0)
wpi	Total	39%	11.1	43296	14.63 (1.8)	7.95 (1.0)	8.56 (1.1)	7.36 (0.9)

example, the burst-1.1 option (pipelining with persistence) for 6-15 objects from the AT&T client took on average 1.40 seconds to retrieve an average of 9.4 objects. This time is approximately 60% (0.6) of the time taken to retrieve the same objects using burst-1.0.

Overall, the burst-1.1 option generally exhibits the best performance with the burst-1.0 and serial-1.1 options in the middle and the burst-1.0 option exhibiting the worst performance. These results are as expected and consistent with those presented in [5] . However, a number of other results come out of examination of the results in Tables 2 and 7:

1. The percentage of servers that exhibit perfect persistence goes down as the number of objects retrieved increases.
2. The relative performance of burst-1.1 (compared to burst-1.0) improves as the number of objects increases. Thus pipelining and persistence improves relative performance with more objects, but also causes more problems in correctly obtaining all objects.
3. The percentage of servers that support perfect persistence is relatively low. Looking at the "Total" row for each client site (or the last column in Table 1), we see the range to be 25-31% of servers. Variations occur because tests were run at different times from different clients under different network and server conditions.

Tables 3 and 8 show the same results except they include all servers that exhibit some persistence. This set of servers includes all perfect persistent servers as well as servers where more than one (but fewer than n) connections are required to retrieve all n objects for at least one of the serial-1.1 and burst-1.1 options. The results show that the relative performance of the serial-1.1 and burst-1.1 options is worse than the perfect persistent sites. This result indicates that the reconnection costs for dropped or lost connections impacts the overall performance to the point that persistence is not as much of an improvement as for perfect persistent servers. Note that the perfect persistent sites still contribute a majority of the results in Tables 3 and 8 so those sites not showing perfect persistence actually have much worse performance than what is shown in the tables.

4.3 Time of Day Analysis

The previous analysis used results from retrievals by a client to a large number of servers. The various protocol options were tested at approximately the same time for each client/server pair, but there was no control when these tests were run. To have more control on when tests were run we created the smaller, select set of servers and created a script to test each server at precise six hour intervals for one week. This script was run from the AT&T, Chile and WPI client sites. A test for the first server in the list was started at 0:02, 6:02, 12:02 and 18:02GMT each day. Tests for subsequent servers in the list were started at three minute intervals for an approximate testing period of three and one-half hours for a single round of testing. Results for each round from each of the three client sites are shown in Table 4, which is of similar format to Tables 2 and 3. Note that the results shown are only for servers exhibiting perfect persistence during at least one of the seven days in each time period.

Table 4: Servers Exhibiting Perfect Persistence Tested at Different Times of Day

Client Site	Time Range (GMT)	Pct. Per-sistent	Ave. Obj. Cnt.	Ave. Obj. Bytes	Ave. Retrieval Time (Sec.)(Rate with Burst-1.0)			
					Serial-1.0	Burst-1.0	Serial-1.1	Burst-1.1
att	00:00-03:30	42%	7.4	32248	1.85 (1.4)	1.30 (1.0)	2.06 (1.6)	1.74 (1.3)
att	06:00-09:30	42%	7.7	33093	1.60 (1.4)	1.18 (1.0)	2.22 (1.9)	1.68 (1.4)
att	12:00-15:30	43%	7.5	32886	3.47 (1.6)	2.22 (1.0)	2.51 (1.1)	2.14 (1.0)
att	18:00-21:30	42%	7.3	31754	3.81 (1.6)	2.31 (1.0)	2.64 (1.1)	2.32 (1.0)
chile	00:00-03:30	32%	9.0	33004	30.80 (1.9)	16.59 (1.0)	17.13 (1.0)	12.98 (0.8)
chile	06:00-09:30	40%	9.1	35322	19.60 (2.0)	9.67 (1.0)	10.80 (1.1)	7.00 (0.7)
chile	12:00-15:30	38%	9.3	35225	25.17 (1.9)	13.11 (1.0)	14.09 (1.1)	9.46 (0.7)
chile	18:00-21:30	35%	9.1	34658	30.09 (1.9)	16.25 (1.0)	17.69 (1.1)	12.86 (0.8)
wpi	00:00-03:30	41%	8.7	33963	16.11 (1.8)	8.76 (1.0)	8.71 (1.0)	6.25 (0.7)
wpi	06:00-09:30	42%	9.0	34889	12.70 (1.9)	6.54 (1.0)	7.09 (1.1)	5.05 (0.8)
wpi	12:00-15:30	43%	9.4	36526	9.28 (1.8)	5.20 (1.0)	5.73 (1.1)	4.04 (0.8)
wpi	18:00-21:30	40%	9.6	37143	22.04 (1.9)	11.56 (1.0)	11.04 (1.0)	8.33 (0.7)

The results show that average performance is generally best for the 6:00-9:30GMT time period (1:00-4:30 on the east coast of the U.S.). WPI results show that the 12:00-15:30 time period is best. Performance is generally the worst for the 18:00-21:30 time period (13:00-16:30 EST). These variations are expected with an approximate ratio of two between the worst and best time periods for a protocol option. Of more interest to our study are the variations in relative performance of the four protocol options. The results show little variation in the relative performance of the options as network/server activity varies.

4.4 Caching

We again used the select set of servers to analyze the end-to-end performance effects of caching and restricted our analysis to those server tests exhibiting perfect persistence. The performance of each of the protocol options is shown in the first row for each client in Table 5. The relative performance of the four protocol options are relatively the same as found in the PROCOW test set for the given number of objects.

Of interest is the second row in the table for each client. This row predicts the performance results if a client cache was used. Because the select data set has a number of tests between the

Table 5: Caching Impact for Servers Exhibiting Perfect Persistence

Client Site	Cache Use	Ave. Retrieval Time (Sec.)(Rate with Burst-1.0)				Ret. Objects	Ret. Bytes
		Serial-1.0	Burst-1.0	Serial-1.1	Burst-1.1		
att	no cache	3.05 (1.7)	1.83 (1.0)	2.39 (1.3)	1.91 (1.0)	7.6	32405
att	with cache	0.51 (1.9)	0.27 (1.0)	0.27 (1.0)	0.28 (1.0)	0.5	6550
att	validate cache	2.44 (1.8)	1.37 (1.0)	1.23 (0.9)	0.74 (0.5)		
chile	no cache	25.88 (1.9)	13.58 (1.0)	14.51 (1.1)	10.32 (0.8)	8.9	33810
chile	with cache	2.91 (1.3)	2.24 (1.0)	2.11 (0.9)	1.63 (0.7)	0.5	6210
chile	validate cache	19.14 (2.0)	9.41 (1.0)	9.92 (1.1)	5.13 (0.5)		
wpi	no cache	13.95 (1.8)	7.54 (1.0)	7.55 (1.0)	5.45 (0.7)	8.9	34873
wpi	with cache	1.35 (1.1)	1.24 (1.0)	1.00 (0.8)	0.80 (0.6)	0.5	6704
wpi	validate cache	10.57 (1.9)	5.65 (1.0)	4.97 (0.9)	2.57 (0.5)		

same client and server we can determine when an object from that server has been previously retrieved. For this study, we assume the cached contents can be reused if the size of the object has not changed. While this assumption is not always valid it is sufficient for the scope of this analysis. The number of objects and bytes retrieved in the presence of a cache is significantly reduced from the results in our test. The results include all of the initial cache misses. These results indicate that the set of Web pages in the select set were relatively static over the week of our study. The performance for each of the protocol options were not measured directly, but derived from the test data with an assumption of zero time for a cache hit. The results show that such a high cache reuse percentage leads to much better performance for all protocol options. As expected, caching has the most relative impact for the slowest serial-1.0 option.

The last row for each client in Table 5 again shows derived costs if each cache hit also incurred a validation cost where the client must send a GET If-Modified-Since or a GET If-None-Match request to the server and receive a 304 response before reusing the cache content. While this assumption is unrealistic, it exaggerates the impact of validation requests on end-to-end performance. Our measured results for an object retrieval differentiate between when the response began and when it was completed. For our derivation we used the time when the response began as an approximation to the time for a header-only 304 response. The results show that the derived validation costs significantly increase the overall time, particularly for the serial-1.0 option. However, the relative performance of serial-1.1 improves relative to the other options.

In summary, the impact of caching is to reduce the number of objects retrieved. We can use the results in Tables 2, 3, 7 and 8 to see that as the number of objects is reduced there is less relative difference between the protocol options. As the tables show, this reduction is not uniform so caching will yield the most cost reduction for serial-1.0 and the least for burst-1.1. The results also show that validation costs can be significant, particularly when connection and request times are in the critical path. As one measure of the validity of our assumption for deriving validation costs, we also issued a "GET If-None-Match: *" request to servers from the AT&T client. The ratio between average If-None-Match and full retrieval times for an object was 0.6. Hence the derived costs for validation shown in Table 5 do not appear unreasonable, but further testing is warranted.

4.5 Multi-Server Content

In analyzing the end-to-end performance impact of embedded objects served by servers other than the base server we use the PROCOW set of servers. The first part of our analysis is to determine the extent to which content is being served from multiple servers. These results are shown in Table

6 where the second column shows the number of cases where the base object includes embedded objects by servers other than the base server. These numbers are relative to the count of base servers in column 2 of Table 1. For example, the HP site shows that 99 out of 665 (15%) base URLs use more than one server to serve content.

Table 6: Use of Multi-Server Content

Client Site	Multi-Server Cnt.	Local Servers Ct.	Local Servers Obj. Pct.	Local Servers Byte Pct.	Ad Servers Ct.	Ad Servers Obj. Pct.	Ad Servers Byte Pct.	Akamai Servers Ct.	Akamai Servers Obj. Pct.	Akamai Servers Byte Pct.	Other Servers Ct.	Other Servers Obj. Pct.	Other Servers Byte Pct.
aciri	97	31	42%	25%	34	11%	3%	10	55%	33%	42	14%	9%
aust	95	32	49%	29%	34	12%	5%	7	54%	40%	38	12%	8%
chile	103	33	39%	20%	34	11%	4%	7	63%	40%	45	13%	7%
hp	99	31	45%	24%	38	11%	4%	11	59%	33%	40	12%	7%
uky	90	27	39%	19%	36	9%	4%	8	69%	43%	39	14%	10%
norway	96	30	44%	27%	36	11%	4%	11	60%	37%	37	16%	9%
safrica	92	28	55%	31%	37	11%	3%	9	53%	38%	39	13%	8%
wpi	92	28	42%	23%	33	12%	7%	8	63%	44%	37	21%	12%

If any part of the content was generated from a server other than the base server, we label such servers as *auxiliary*. To further explore these results, we classified each of the auxiliary servers used. We did this classification through a combination of looking at the auxiliary server's name and IP address (results from the AT&T client are not shown because of problems due to masking when the IP address is printed). If the network portion of auxiliary server IP address matched the network portion of the base servers address it was classified as a *local* server (i.e., local to the *base* server, not the client). We realize that such a classification may not always be correct but we were able to verify by hand most of the sample. Non-local servers containing the string "ad" in the server name (such as *adforce*) were classified as ad servers. Non-local servers containing the string "akamai" were another category. Akamai [14] is a commercial company, which serves content for contracted sites. All other non-local servers were grouped in the final category as *Other*. The purpose of these other sites is unknown—some may also serve ads, count clicks, or serve contracted content.

Results in Table 6 show the count of non-local servers in each category as well as the percentage of objects and bytes served by this category relative to the total number of objects and bytes for the base Web page. The percentages only include Web pages where the given category of servers are present. All cases show that auxiliary servers serve relatively more objects than bytes.

The second part of our study was to examine the performance impact of these categories on the end-to-end performance. For this analysis we determined the rate at which objects and bytes are served from each category of server. Rather than restrict ourselves to sites exhibiting persistence we used the best case time of the supported protocol options for each server. Servers in the same network as base servers (we call them *local* servers) have a higher object rate but a lower byte rate. Ad servers are mixed on object rate and lower on byte rate. Some of the servers that we categorized as others could be ad servers as well. Their byte rate is often less than the base servers. The results also show that content distribution servers (which is Akamai in our test set) improve performance for the most part (ranging from 0.7 to 7.5 when compared to throughput from the "base" servers). The Akamai servers are not necessarily faster due to geographical proximity—clients in South Africa and Norway use Akamai servers in New York, San Diego, and Boston. We note that Akamai servers are being used in a tiny fraction (less than 1.5%) of the sites we tested.

4.6 Range Requests

We examine a subset of requests sent to HTTP/1.1 servers that were able to handle `Range` requests to see if there was appreciable reduction in latency in getting just the first hundred bytes and the first thousand bytes of a resource as compared to a full retrieval. For the first hundred bytes test, among the servers that responded correctly with the 206 Partial Content response, the average latency for a `Range` response was 60% that of a full response for the AT&T Labs data, though the average full response time was only half a second. The numbers for South Africa are 53% with average full response time of 3.84 seconds, Norway 65% with average response time 0.72 seconds. For the thousand bytes range test the numbers are similar (AT&T 58.68% with the average full response time of 0.71 seconds, South Africa 56.59% with 3.84 seconds average, Norway 58.78% with 1 second average). So it is clear that there is improvement in user perceived latency but the appreciable nature of improvement depends on the speed of the link from a client site.

5 Discussion

Our results show that the HTTP/1.1 protocol is indeed an improvement in some sense. Two limitations of our testing: not all the servers being tested exhibit all the features of HTTP/1.1 and we only test a subset of the features, makes it harder to present an overall evaluation of the improvements. However, we believe that our work is a step in the right direction. Our global testing infrastructure is solidifying and our process is largely automated. We expect more content diversification on the Web and the dependence on DNS for such methods needs to be examined more closely.

The absence of a policy on when to close persistent connections in the HTTP/1.1 protocol standard is understandable. However this has led to significant variance in holding times. While there is not a clear algorithm, it is difficult to get a precise idea by remote tests when servers use different criteria for deciding when to close a persistent connection. It might be beneficial for sites to decide for themselves how many requests they should handle on a single persistent connection. There is however no doubt that persistent connections with pipelining significantly reduces latency from a client perspective when supported.

6 Related Work

Outside the Web context, there have been a number of studies to measure end-to-end performance of the Internet. Paxson has done a number of studies on this issue [15] .

There have been many studies to examine Web performance from various perspectives. Nielsen, et al published the first work on measured performance of the HTTP/1.1 protocol [5] . A related piece of work studied the performance interactions of persistent HTTP with TCP. Both of these studies found significant interaction problems that needed correction before persistent HTTP performed as expected. These works build on prior work to examine the impact of persistent connection HTTP [16] .

More recent work has examined how bottlenecks in the network, CPU and the disk system affect the relative performance of HTTP/1.0 and HTTP/1.1 [17] . This work describes a controlled experiment to understand what relative impact of these subsystems on the protocol versions. One of the results of this work is a recommendation that HTTP/1.1 clients implement an "early close" policy where the client closes a connection after retrieving all objects associated with a Web page. These same authors published a recent article on instrumenting clients, servers and various points in the Internet to perform an integrated server/network measurement [18] .

A number of other studies and tools directly examine the user-perceived performance of the Web. Keynote makes available a tool to visualize the performance of a Web retrieval [19] . Kruse, et al examine the impact of interleaving requests on user-perceived performance [20] . Gilbert and Brodersen demonstrate the benefits of progressive delivery of Web images in Web page rendering [21] .

7 Summary and Future Work

In this work, we sought to examine the components of end-to-end delay in a user's Web experience by performing a large scale study of popular Web sites. We have built on the PROCOW infrastructure by adding additional clients to the collection of client sites around the world. We have presented results on performance improvements due to the changes in the HTTP/1.1 protocol and some preliminary notions of impact of caching and content distribution via multiple servers.

The next stage of our experiment is examination of the contribution of network aspects of the latency (DNS, Redirections, TCP) and see how they interact with the HTTP layer. This includes passive measurements (e.g., *tcpdump*) to examine the variance against the observed measurements at the application layer. We also plan to examine a different set of target servers that are responsible for a larger portion of the bandwidth usage in terms of response bytes as opposed to just request popularity.

8 Acknowledgements

The authors would like to thank several people who were helpful in getting us access to machines in many parts of the world—the study would not be possible without their help. They include Martin Arlitt, Alan Barrett, Steven Bellovin, Randy Bush, Jim Griffioen, Eduardo Krell, Anders Lund, Mark Murray, Scott Shenker, Graeme Yates. We thank Mikhail Mikhailov for assistance in setting up the testing framework. We thank Bruce Maggs for answering questions related to Akamai.

Balachander Krishnamurthy

FP-HA1630000-BK/CEW **Craig E. Wills**

Copy to
Members of HA163
Members of HA171
Members of HA173
Hamid Ahmadi
Lee Breslau
Dir. HA16

Table 7: Servers Exhibiting Perfect Persistence from Additional Client Sites

Client Site	Object Count Range	Pct. Per-sistent	Ave. Obj. Cnt.	Ave. Obj. Bytes	Ave. Retrieval Time (Sec.)(Rate with Burst-1.0)			
					Serial-1.0	Burst-1.0	Serial-1.1	Burst-1.1
aciri	2-5	51%	3.2	16784	2.18 (1.0)	2.17 (1.0)	1.79 (0.8)	1.90 (0.9)
aciri	6-15	29%	9.8	52408	5.40 (2.7)	2.00 (1.0)	2.15 (1.1)	1.26 (0.6)
aust	2-5	47%	3.2	16143	10.61 (1.4)	7.42 (1.0)	7.10 (1.0)	6.16 (0.8)
aust	6-15	26%	9.7	49287	27.28 (2.4)	11.40 (1.0)	13.24 (1.2)	10.93 (1.0)
aust	16+	18%	24.3	82522	59.80 (2.8)	21.06 (1.0)	28.11 (1.3)	19.20 (0.9)
aust	Total	30%	9.3	38258	25.11 (2.2)	11.29 (1.0)	13.06 (1.2)	10.09 (0.9)
aciri	16+	20%	25.4	82370	9.64 (2.7)	3.59 (1.0)	5.21 (1.5)	3.05 (0.9)
aciri	Total	33%	9.8	39956	4.62 (1.9)	2.45 (1.0)	2.64 (1.1)	2.01 (0.8)
hp	2-5	48%	3.3	16110	1.61 (1.3)	1.20 (1.0)	1.69 (1.4)	1.37 (1.1)
hp	6-15	26%	9.8	53839	5.18 (2.2)	2.32 (1.0)	2.55 (1.1)	1.52 (0.7)
hp	16+	17%	24.5	82042	10.22 (2.5)	4.07 (1.0)	3.80 (0.9)	2.45 (0.6)
hp	Total	30%	9.2	38756	4.24 (2.1)	2.06 (1.0)	2.33 (1.1)	1.63 (0.8)
uky	2-5	47%	3.2	18195	6.64 (1.3)	5.07 (1.0)	4.27 (0.8)	3.59 (0.7)
uky	6-15	28%	9.7	54820	14.00 (2.4)	5.80 (1.0)	5.62 (1.0)	3.88 (0.7)
uky	16+	18%	25.4	83219	38.76 (2.8)	13.69 (1.0)	15.69 (1.1)	7.01 (0.5)
uky	Total	31%	9.7	41617	15.53 (2.2)	7.15 (1.0)	7.11 (1.0)	4.41 (0.6)
norway	2-5	47%	3.1	15829	3.50 (1.6)	2.19 (1.0)	2.65 (1.2)	2.23 (1.0)
norway	6-15	8%	9.9	30000	8.00 (2.1)	3.79 (1.0)	3.97 (1.0)	2.63 (0.7)
norway	16+	2%	26.3	79655	19.18 (2.9)	6.57 (1.0)	11.24 (1.7)	8.09 (1.2)
norway	Total	19%	4.7	19595	4.55 (1.8)	2.53 (1.0)	3.09 (1.2)	2.46 (1.0)
safrica	2-5	47%	3.3	16724	14.22 (1.4)	10.45 (1.0)	10.35 (1.0)	10.71 (1.0)
safrica	6-15	25%	9.4	40670	35.43 (2.1)	16.95 (1.0)	22.36 (1.3)	17.20 (1.0)
safrica	16+	16%	25.2	75905	83.90 (2.8)	29.79 (1.0)	47.13 (1.6)	30.28 (1.0)
safrica	Total	30%	9.0	33994	32.90 (2.1)	15.78 (1.0)	20.40 (1.3)	16.08 (1.0)

Table 8: Servers Exhibiting Some Persistence from Additional Client Sites

Client Site	Object Count Range	Pct. Per-sistent	Ave. Obj. Cnt.	Ave. Obj. Bytes	Ave. Retrieval Time (Sec.)(Rate with Burst-1.0)			
					Serial-1.0	Burst-1.0	Serial-1.1	Burst-1.1
aciri	2-5	56%	3.3	17626	2.23 (1.0)	2.25 (1.0)	1.97 (0.9)	2.09 (0.9)
aciri	6-15	45%	10.2	53999	5.79 (2.3)	2.48 (1.0)	2.67 (1.1)	2.46 (1.0)
aciri	16+	33%	24.9	80486	8.69 (2.7)	3.24 (1.0)	4.69 (1.4)	3.21 (1.0)
aciri	Total	44%	11.1	45027	5.00 (1.9)	2.59 (1.0)	2.91 (1.1)	2.50 (1.0)
aust	2-5	55%	3.3	17934	10.68 (1.4)	7.44 (1.0)	7.31 (1.0)	6.72 (0.9)
aust	6-15	36%	9.9	48284	31.05 (2.5)	12.42 (1.0)	16.15 (1.3)	13.65 (1.1)
aust	16+	25%	25.4	82561	60.28 (2.9)	20.76 (1.0)	29.78 (1.4)	19.93 (1.0)
aust	Total	39%	10.3	41186	27.76 (2.3)	11.90 (1.0)	14.95 (1.3)	11.66 (1.0)
hp	2-5	54%	3.3	17397	1.62 (1.3)	1.29 (1.0)	1.86 (1.4)	1.56 (1.2)
hp	6-15	39%	10.1	51701	5.88 (2.2)	2.73 (1.0)	2.96 (1.1)	2.60 (1.0)
hp	16+	31%	25.5	78306	10.09 (2.5)	4.11 (1.0)	4.04 (1.0)	3.79 (0.9)
hp	Total	41%	11.4	43653	5.14 (2.1)	2.46 (1.0)	2.76 (1.1)	2.47 (1.0)
uky	2-5	54%	3.3	18314	6.53 (1.4)	4.83 (1.0)	4.13 (0.9)	4.03 (0.8)
uky	6-15	44%	10.2	52999	16.10 (2.0)	8.05 (1.0)	8.22 (1.0)	6.41 (0.8)
uky	16+	35%	25.0	81353	38.75 (2.6)	14.64 (1.0)	15.38 (1.1)	9.75 (0.7)
uky	Total	44%	11.6	46663	18.75 (2.2)	8.63 (1.0)	8.61 (1.0)	6.39 (0.7)
norway	2-5	52%	3.2	17162	3.63 (1.7)	2.18 (1.0)	2.83 (1.3)	3.04 (1.4)
norway	6-15	17%	9.4	48143	11.02 (2.1)	5.13 (1.0)	5.47 (1.1)	10.65 (2.1)
norway	16+	6%	24.6	63839	16.12 (2.3)	7.06 (1.0)	9.70 (1.4)	14.31 (2.0)
norway	Total	25%	6.2	27257	6.15 (1.9)	3.18 (1.0)	3.93 (1.2)	5.51 (1.7)
safrica	2-5	53%	3.3	17851	15.81 (1.4)	11.52 (1.0)	12.23 (1.1)	12.24 (1.1)
safrica	6-15	41%	10.4	46758	41.52 (2.0)	20.72 (1.0)	29.05 (1.4)	22.04 (1.1)
safrica	16+	36%	25.4	79928	84.20 (2.6)	32.46 (1.0)	53.23 (1.6)	39.37 (1.2)
safrica	Total	44%	11.9	44300	43.30 (2.1)	20.29 (1.0)	29.09 (1.4)	23.04 (1.1)

References

[1] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, HTTP Working Group, June 1999. `ftp://ftp.ietf.org/rfc2616.txt`.

[2] B. Krishnamurthy and M. Arlitt, "PRO-COW: Protocol compliance on the web," Tech. Rep. 990803-05-TM, AT&T Labs, August 1999. `http://www.research.att.com/~bala/papers/procow-1.ps.gz`.

[3] C. E. Wills and M. Mikhailov, "Towards a better understanding of web resources and server responses for improved caching," in *Eighth International World Wide Web Conference*, (Toronto, Canada), May 1999. `http://www.cs.wpi.edu/~cew/papers/www8.ps.gz`.

[4] B. Krishnamurthy, J. C. Mogul, and D. M. Kristol, "Key differences between HTTP/1.0 and HTTP/1.1," in *Eighth International World Wide Web Conference*, (Toronto, Canada), May 1999. `http://www.research.att.com/~bala/papers/h0vh1.ps.gz`.

[5] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. Lie, and C. Lilley, "Network performance effects of HTTP/1.1, CSS1, and PNG," in *Proceedings of the ACM SIGCOMM '97 Conference*, ACM, Sept. 1997. `http://www.acm.org/sigcomm/sigcomm97/papers/p102.html`.

[6] P. Cao and S. Irani, "Cost-aware WWW proxy caching algorithms," in *Symposium on Internet Technology and Systems*, USENIX Association, Dec. 1997. `http://www.usenix.org/publications/library/proceedings/usits97/cao.html`.

[7] B. Krishnamurthy and C. E. Wills, "Piggyback server invalidation for proxy cache coherency," in *Seventh International World Wide Web Conference*, (Brisbane, Australia), pp. 185–193, Apr. 1998. Published in Computer Networks and ISDN Systems (30)1-7 (1998) pp. 185–193. `http://www.cs.wpi.edu/~cew/papers/www7/www7.html`.

[8] "Media metrix." `http://www.mediametrix.com`.

[9] "The netcraft web server survey." `http://netcraft.co.uk/survey/`.

[10] "100hot.com." `http://www.100hot.com`.

[11] "1999 Fortune 500 companies," Fortune volume 139 number 8, April 26 1999.

[12] "1998 Global 500 companies," Fortune Magazine 1998.

[13] D. Mosberger and T. Jin, "httperf – a tool for measuring web server performance," in *Workshop on Internet Server Performance*, (Madison, Wisconsin USA), June 1998. `http://www.cs.wisc.edu/~cao/WISP98/final-versions/davidm.ps`.

[14] "Akamai." `http://www.akamai.com`.

[15] V. Paxson, "End-to-end internet packet dynamics," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 277–292, June 1999. `ftp://ftp.ee.lbl.gov/papers/vp-pkt-dyn-ton99.ps.gz`.

[16] J. C. Mogul, "The case for persistent-connection HTTP," in *Proceedings of the ACM SIGCOMM '95 Conference*, ACM, Aug. 1995. `http://www.acm.org/sigcomm/sigcomm95/papers/mogul.html`.

[17] P. Barford and M. Crovella, "A performance evaluation of hyper text transfer protocols," in *Proceedings of the ACM SIGMETRICS '99 Conference*, (Atlanta, Georgia), ACM, May 1999.

[18] P. Barford and M. Crovella, "Measuring web performance in the wide area," *ACM Performance Evaluation Review*, vol. 27, pp. 37–48, Aug. 1999.

[19] "Keynote lifeline."
`http://lifeline.keynote.com/Lifeline/buyitonline/snapshot.asp`.

[20] H. Kruse, M. Allman, and P. Mallasch, "Network and user-perceived performance of web page retrievals," in *Proceedings of the First International Conference on Telecommunications and Electronic Commerce*, (Nashville, TN USA), November 1998.
`http://roland.lerc.nasa.gov/~mallman/papers/ecom98.ps`.

[21] J. Gilbert and R. Brodersen, "Globally progressive interactive web delivery," in *Proceedings of the IEEE Infocom '99 Conference*, (New York, NY), IEEE, Mar. 1999.

Document Cover Sheet for Technical Memorandum

Title: Analyzing factors that influence end-to-end Web performance

Authors	Electronic Address	Location	Phone	Company (if other than AT&T)
Balachander Krishnamurthy	bala@research.att.com	FP D229	973-360-8648	
Craig E. Wills	cew@cs.wpi.edu	Worcester MA		Worcester Polytechnic Institute

Document No.	Work Project No.
HA1630000-991129-07TM	539054-0000

Keywords:
Web Performance, Web Protocols, End-to-End Performance, Active Measurement

MERCURY Announcement Bulletin Sections
CMP– Computing

Abstract

With the growing popularity of the Web, performance issues have come to the fore both on an intranet basis and on an Internet basis. Web performance impacts the popularity of a particular Web site or service as well as the load on the network. While Web caching has become popular and has been studied extensively, there have been no publically available *end-to-end* measurements that have focused on a large number of popular Web servers examining the components of delay or the effectiveness of the recent changes to the HTTP protocol. In this paper we report on an extensive study carried out from several client sites geographically distributed around the world to a collection of over 700 servers to which a majority of Web traffic is directed. Our results show that the primary changes in the recently draft standardized HTTP/1.1 protocol are indeed useful but further examination is warranted.

Pages of Text 14 Other Pages 6 Total 20
No. Figs. 0 No. Tables 8 No. Refs. 21

Mailing Label

Complete Copy	Cover Sheet Only
Members of HA163	Larry Rabiner
Members of HA171	Members of HA16
Members of HA173	Members of HA17
Hamid Ahmadi	DHs HA16
Lee Breslau	
Dir. HA16	

Future AT&T Distribution by ITDS Release to any AT&T employee (excluding contract employees)

Author Signatures

Balachander Krishnamurthy Craig E. Wills

For Use by Recipient of Cover Sheet:

Computing network users may order copies via `http://attlis.att.com/services/proprietary.html`.