In *Performance Evaluation Review*
Special Issue on Network Traffic Measurement and Workload Characterization, August 1999

Measuring Web Performance in the Wide Area

Paul Barford and Mark Crovella
Computer Science Department
Boston University
111 Cummington St, Boston, MA 02215
{barford,crovella}@cs.bu.edu

Abstract

One of the most vexing questions facing researchers interested in the World Wide Web is why users often experience long delays in document retrieval. The Internet's size, complexity, and continued growth make this a difficult question to answer. We describe the Wide Area Web Measurement project (WAWM) which uses an infrastructure distributed across the Internet to study Web performance. The infrastructure enables simultaneous measurements of Web client performance, network performance and Web server performance. The infrastructure uses a Web traffic generator to create representative workloads on servers, and both active and passive tools to measure performance characteristics. Initial results based on a prototype installation of the infrastructure are presented in this paper.

1 Introduction

The World Wide Web is a fascinating example of a distributed system on an immense scale. Its large scale and increasingly important role in society make it an important object of study and evaluation. At the same time, since the Web is not centrally planned or configured, many basic questions about its nature are still open.

The question that motivates our work is a basic one: why is the Web so slow? More precisely: what are the root causes of long response times in the Web? Our goal is to answer this question as precisely as possible, while recognizing that it does not admit to a single simple answer.

Attempting to answer this question immediately exposes gaps in the research to date on Internet measurement. To answer this question, two research needs must be addressed:

1. The need for *integrated* server/network measurement. Most related research to date has focused on measuring servers and networks in isolation; the interactions between the two, especially in a wide-area setting, are not well understood.

2. The need to understand the relationship between *active* and *passive* measurements in the Internet. Active measurements are easily understood, but do not clearly predict actual Web performance; passive Web measurements can reflect actual performance but can be hard to interpret.

This paper describes a project framework, called *Wide Area Web Measurement* (WAWM), intended to meet these two needs. The project employs both network and server measurement, and as well uses both passive and active measurement approaches. Our goal in this paper is to describe the project, to explain its place in the context of Internet measurement research, and to demonstrate the utility of our approach using samples of our early measurements.

Our measurement architecture is based on measuring server state *and* network state simultaneously; on taking concurrent measurements at both the client and the server ends of sample connections; and on varying server load so as to allow controlled exploration of a wide range of server/network interactions. For example, by varying server load and running experiments at different times of day, we can explore the four cases in which the server and the wide-area network are independently either heavily or lightly loaded (in qualitative terms).

The test apparatus decribed in this paper is a small-scale prototype intended to expose design issues and act as a platform for developing test infrastructure and expertise. It consists of a Web server running at our site, along with a number of local clients capable of generating varying load on the server (even up to an overloaded condition). A single off-site client, located across the Internet (15 hops) from us, generates test connections to our server. Passive packet trace measurements are taken both at the off-site client and at the server. Active measurements of the connection path are also collected during each transfer using Poisson Ping (`Poip`) [1] which sends UDP packets along the same path as the connection and measures the time-averaged network state (packet delay and loss rate). Although this apparatus is a reduced version of the full-scale system we will eventually deploy (as described in Section 3), it is sufficient to yield surprising and informative results.

The results we present in Section 4 are only samples and do not yet represent a thorough assessment of wide-area Web performance. Nonetheless they yield some immediate insights about the benefits of integrated server/network measurement, and about the relationship between active and passive network measurement.

For example, we find that server load has distinctive effects on the pattern of packet flow through the network. We show that for our experimental setup, one of the most noticeable effects of high server load is to generate a significant gap between the transmission of connection setup packets and the first data packet flowing from the server. While this gap can be understood as a consequence of socket and file operations, it is notable that for typical (*i.e.,* short) transfers (20 KB) the gap generally *dominates* transfer time.

In addition, we show an even more surprising effect of server load. When the network is heavily loaded (*i.e.,* packet loss rates are high), it is not uncommon for a heavily loaded server to show *better* mean response time than a lightly loaded server. Our measurements suggest that this may be because heavily loaded servers often show lower packet loss rates, and since packet losses have dramatic effects on transfer latency, this can reduce mean response time.

Finally, we also explore the relationship between active and passive measurements. We show that `Poip` measurements of packet delay are generally lower than those experienced by packets in TCP connections; this effect may be due to the fact that TCP packets flow in a burstier pattern than do `Poip` packets. We also show that `Poip` measurements of packet loss are not strongly predictive of packet loss experienced by TCP connections, which may be due to the feedback nature of TCP in the presence of loss.

These results demonstrate the utility of integrated network/server measurement. We conclude that although the answers are not yet in hand, our approach appears to show promise for addressing the motivating question: Why is the Web so slow?

2 Related Work

The WAWM project is based on a large and diverse body of prior work. In particular, we rely on results from Web characterization studies, passive and active network measurement studies, time calibration studies, and wide area network performance studies. In this section we present selections of the studies from each of these areas which served as the basis for our work and contrast that work with the goals of the WAWM project.

2.1 Web Performance Characterization

The body of prior work on measurements and characterizations of the Web is too large to survey here. That work relates to ours in providing guidance in selection of workloads. We use the SURGE workload generator [6] to generate representative local loads on our server.

Web server behavior has been studied extensively [3, 4, 7, 14, 22, 30]. These studies provide insight into server behavior under various loads and versions of HTTP, and provide us with methodologies for measuring performance in both the user space and system space.

Studies of the network effects of Web traffic include [5, 13, 19, 21, 29, 31]. These studies show the performance effects of both HTTP protocol interaction and TCP packet level interactions. The shortfall of these studies when it comes to analyzing latency is that they only take measurements in one place (somewhere near an end point of an HTTP transaction) and, thus, cannot provide insight into the packet delays in both directions of a transaction.

2.2 Passive and Active Network Measurements

The task of assessing general network performance has been well studied. A good source of general information on current measurement tools and projects can be found at [45]. General techniques for studying network performance can be split into those which use passive monitoring techniques and those which use active techniques. Passive techniques typically consist of gathering packet level data at some tap point in the network. Examples of passive monitors include `coral` [33] and `tcpdump` [46]. A great deal of information can be extracted from passive monitoring, even at only a single point. One of the benefits of passive techniques is that their use does not perturb the system being measured. However, passive monitoring has limitations in terms of the amount of data which is gathered during tests (typically very large) and the difficulty of real time analysis of the data.

Active measurement tools inject some kind of stimulus (a packet or series of packets) into the network and then measure the response to that stimulus. Active measurement tools can be used to measure bulk throughput, path characteristics, packet delay, packet loss, and bandwidth characteristics between hosts.

There are a number of tools which can be used to measure end-to-end delay and packet loss; we currently use Poisson Ping (`Poip`) [1] in the WAWM project. `Poip` injects UDP packets into the network according to a Poisson process. The motivation for this method is that sampling at intervals determined by a Poisson process will result in observations that match the time-averaged state of the system (assuming it is in a stationary state). There are a number of studies that have used similar active measurements to study packet loss including [8, 48] and to drive models of TCP behavior including [24, 34]. These studies provide background that guides our active measurement approach.

A number of active measurement tools have been developed that can be used to measure bandwidth along the end-to-end path including `pathchar` [18], `bprobe` [10], and `nettimer` [20]. Traceroute [17] is an active measurement tool that gathers simple path characteristics. TReno [23] or "Traceroute Reno" is a tool used to measure bulk TCP throughput capacity over an Internet path. These active measurements can give insight into network conditions, but depending on how much data they inject into the network, they can also perturb the network's performance. In the WAWM project we compare active and passive measurements in order to understand how active measurements of network conditions correlate to performance as seen at the client and at the server.

Some of the most extensive work using both passive and active measurement techniques has been done by Paxson in a series of studies reported in [35, 36, 37, 39]. These studies are a starting point for the WAWM project from both measurement and analysis perspectives.

2.3 Time Calibration

Measurements of end-to-end delay in our architecture are based on having nearly synchronized clocks on the client and the server. The difficulties of synchronizing clocks in a widely distributed environment have been well studied, especially by Mills [27, 28]. That work led to the development of the network time protocol (NTP) daemon which we use in WAWM to synchronize the clocks on clients and servers. NTP assures that clocks are synchronized on the order of tens of milliseconds. Additional studies of clock synchronization problems include [32, 40]. Paxson points out in [40] that the use of NTP provides close synchronization of clocks but that checks for skew should still be made when analyzing data. The particular results reported in this paper are not strongly affected by clock skew, however in future work we plan to either correct for clock skew or use a synchronized external time source (GPS).

2.4 Wide Area Network Performance Measurement

There are a number of wide area measurement projects which are either proposed or underway at the present time. None propose to do the type of work that is being proposed for WAWM; however there are some similarities. The projects which are the most closely related to WAWM include: the Cooperative Association for Internet Data Analysis (CAIDA) [12], the National Internet Measurement Infrastructure (NIMI) project [1], IETF's Internet Protocol Performance Metrics (IPPM) working group [26, 41], NLANR's Measurement and Operations Analysis Team (MOATS) project [11], the Internet Performance Measurement and Analysis project (IPMA) [25], Keynote Systems, Inc. [15], and the Surveyor Project [43]. Each of these projects attempts to measure or analyze some aspect of Internet performance. The WAWM project is similar to the above projects in that it uses a wide area measurement infrastructure and that there are significant data management and analysis tasks. The WAWM project is different from these for a number of reasons. First, WAWM project focuses on a *user level* application — the Web. Most of the projects listed above focus on general network measurement, monitoring and analysis. Although WAWM proposes to take network level measurements during tests, measurements of user level characteristics will be made at the same time. Second, WAWM proposes to *install* the systems which will act as servers in the infrastructure. This approach facilitates ease and consistency of software installation, control over the environment, security, and ease of management. Third, the systems which will be acting as servers in the WAWM tests will also have performance monitoring tools which will enable detailed system measurements during tests.

3 WAWM Project Overview

In this section we describe the tools and methods used in the WAWM project. We break them down into four categories: measurement architecture and tools, management tools, testing protocol, and analysis methods. In each category we first describe the long-term project goals, then we describe the specific implementations used in the results reported here.

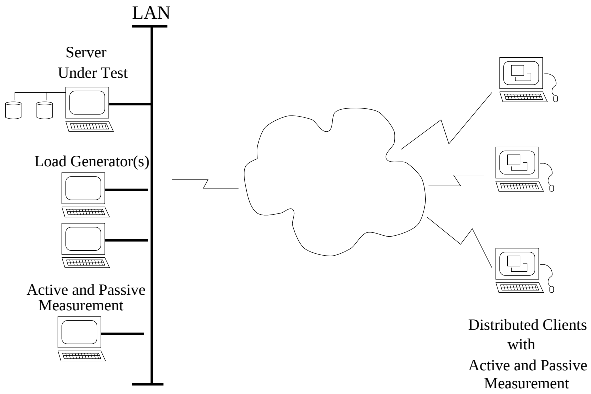


Figure 1: General Architecture of WAWM Hardware Components

3.1 Measurement Architecture

3.1.1 General Architecture

The general architecture of the WAWM hardware components is shown in Figure 1. On the left are components that are located at our site. The first system is configured as the Web server and is only used to serve documents to the clients. Servers are also configured with performance monitoring tools so that detailed measurements of CPU, memory and network characteristics can be made during tests. The second and third systems are configured as local load generators. They run the SURGE workload generator which is used to adjust the amount of load on the server throughout the duration of a test. This is important since we cannot expect remote systems to generate enough load to saturate the server when we want to run tests under heavy server load. The fourth system is used to take both passive measurements (*e.g.,* packet traces) and active measurements (*e.g.,* using network probe packets). Using these tools we expect to measure (at minimum) packet delays, packet losses, path routes, and path characteristics such as bottleneck link speed. Another system (not shown) is used as the control system which initiates tests and gathers data from the infrastructure. All of the systems in the server cluster are connected to the local Internet service provider (ISP) such that the server cluster network traffic is not a bottleneck.

On the right side of the figure are systems that are located remotely from our site. Our goal is to distribute these systems widely across the Internet in an attempt to explore a wide range of path types. The remote systems make requests for files from the server during tests. They are also configured with both passive and active network measurement tools. Requests are generated by clients using a modified version of SURGE. This modified version can be configured to run in its standard mode or to make repeated requests for a single file in much the same way as WebStone [47].

We are influenced by Paxson's *Network Probe Daemon* study [39] in terms of the number of remote sites we would like to have in the complete WAWM. That study utilized 37 sites which enabled measurement of over 1,000 distinct Internet paths. Selection of the additional sites will be based on geographic location and Internet connection type with heterogeneity as the goal. At least one additional server site will also be included to diversify measurements.

3.1.2 Initial Prototype

We have implemented a prototype of this architecture in order to expose measurement and design issues and act as a platform for developing test infrastructure expertise. The results in this paper are all taken from this prototype system. In our prototype, a single server cluster was located at Boston University (BU) and consisted of six PCs connected via switched 100Mbps Ethernet which is connected to the BU ISP via a 10Mbps bridge. The PCs were configured with 200Mhz Pentium Pro CPUs and 128MB of RAM each. Each system ran Linux v2.0.30. The server was configured with Apache 1.3.0 [42]. `Traceroute` v1.4a5 and `tcpdump` v3.4 were used to measure route characteristics and gather packet traces respectively. `xntpd` v3-5.93 was used to synchronize time stamps between the server cluster and the remote system. One of the PCs in the server cluster was set up as the time server for all tests. The `xntpd` update interval was set to 64 seconds. `Poip` was configured to run both to and from the server cluster with the mean measurement interval set to one second and with a packet size of 256 bytes. The system set up as the Web server was also configured with Webmonitor [2] to measure detailed server performance characteristics in the CPU, memory and network components of the server.

The local load generating systems in the server cluster were configured with SURGE which was set up to make requests for a file set which consisted of 2000 distinct files. The total size of this data set was about 50MB; thus the entire data set could be cached in the server's RAM. Experiments were run under either *low* or *high* local load levels. Load levels in SURGE are defined in terms of *user equivalents* (UEs). Characterizations of load under a range of UEs for the same PC systems are presented in [7]. Using HTTP/1.0, we used 40 UEs for *low* load and 520 UEs for *high* load.

The prototype installation used only a single remote site which was located at the University of Denver (DU). This system was configured with a 333Mhz Pentium Pro CPU and 64MB of RAM and ran Linux v2.0.35. It was connected in the local area via switched 100Mbps Ethernet and then to the DU ISP via a 10Mbps bridge. The system was set up with the same network measurement and time synchronization tools as the systems in the server cluster. The modified version of SURGE was set up to make repeated requests for one of three files on the server. The three file sizes selected were 1KB, 20KB and 500KB. These file sizes were chosen to demonstrate a range of behavior in the network: 1KB files measure essentially only the connection setup time; 20KB files are typical-sized Web transfers and typically involve about 6 or 7 round trips, so are dominated by TCP's slow-start behavior; 500KB files allow us to observe the mode in which most packets are sent in the Web, namely in TCP's congestion avoidance mode.

3.2 Management Architecture

In order to run tests, download data and manage systems which are part of the WAWM infrastructure, a secure management platform is required on all WAWM systems. We are evaluating third-party management platforms including the NIMI platform and INSoft's VitalAgent [16]. Both provide the essential management features and flexibility necessary for WAWM as well as built-in network performance measurement tools.

The management platform is an area of the system in which practical matters can dictate design choices. Therefore, in order to gain insight into the relevant issues, we developed our own management tool for the prototype study. The tool is written in Perl and is based on using Secure Shell [44] (SSH v1.2.26 was used in the prototype tests). This tool proved to be sufficient for the prototype tests.

The data collected is cataloged and maintained in such a way as to facilitate analysis and reuse by other researchers. The WAWM data repository is maintained at the same central site from which WAWM tests are run. At this point in time a simple file structure is adequate to store the data. Each test run in our prototype study with only a single client resulted in about 9MB of compressed results data.

3.3 Testing Protocol

3.3.1 General Architecture

A WAWM *test* is a period of time over which a specific set of client systems make requests to a specific server. Specification of tests requires determining the following:

1. The server cluster which will be used in the test.

2. The local load which will be placed on the server.

3. The set of remote clients which will make requests during a test. The influence of distance from the server, Internet connection type and the number of clients participating in a test will all influence the outcome of measurements.

4. The time of day the test will be run. Due to the diurnal cycle of Internet traffic in North America, tests run during the day typically experience more pronounced influence from cross traffic than those run late at night.

5. The duration of the test. The test must be run long enough to collect a representative sample of data for the given network and server conditions.

6. The schedule for active network measurements during tests. Since active measurements could perturb HTTP performance, this schedule must be selected such that the resulting data gives a clear picture of the network's state while not influencing the HTTP tests.

3.3.2 Initial Prototype

The testing protocol used in the prototype ran one set of tests during the day (between 11:00am and 7:00pm) and one set of tests at night (between

9:30pm and 5:00am) for each of ten days. Initiating tests at these times provided a variety of network conditions for our data. Each test began by running `traceroute` to and from the server which gave us an indication of the current end-to-end routes during the tests. Each test was run for 15 minutes; we observed that this duration was always sufficient to transfer at least 40 of the 500KB files under heavy network load conditions and 4696 of the 1KB files under light network load conditions. `Poip` was run both to and from the server simultaneously with the HTTP requests for the duration of each test. Six separate tests were run during each test set. They consisted of making repeated requests for a single file (1KB, 20KB or 500KB) under both low and high server loads.

3.4 Analysis Protocol

3.4.1 General Architecture

Once test data is collected, it is reduced and analyzed. Automation of the analysis process is necessary due to the large amount of data that is gathered during each test. We have begun the development of a tool called `tcpeval` which will perform these tasks on the packet trace data. Our model is based on `tcpanaly` [38]. In addition to statistical approaches, we intend to develop methods for detailed analysis of individual file transfers. Detailed analysis may require a visual "coordination tool" much like the tool developed in [9].

We also use `natalie` [1] to analyze `Poip` traces and intend to develop methods to correlate the data gathered through active and passive measurements.

3.4.2 Initial Prototype

At the highest level we currently tabulate summary statistics and distributional statistics of file transfer delays, packet delays, packet loss and server performance. `tcpeval` also generates a number of graphical aids to interpretation, including timeline diagrams which illustrate individual packet exchanges between a client and the server within a single HTTP transaction. In this paper we focus on timeline diagrams (*e.g.,* Figures 3, 5, and 7) because they can represent the timing of events at both ends of a connection.

4 Sample Results

In this section we report on initial results obtained using the prototype apparatus described in Section 3. The testing protocol called for `traceroute` to be run before and after each test. These measurements showed that the route to and from the remote site was asymmetric but fairly stable for all of the tests (13 hops from BU to DU and 15 hops from DU to BU). While there were occasional route changes between sets of test, we measured only one route change *during* one of the tests.

4.1 Measurements of file transfer latency

First we present performance measurements for file transfers over combinations of server load, network load and file sizes used in the tests. While measurements were taken over a two week period, we present results of data sets which are typical of all of the data that was collected.

4.1.1 Performance under light network load

The details of the file transfer latency under light network load for all three files can be seen in Table 1. The table shows that the busy server adds approximately one second to the average file transfer latency for the 1KB and 20KB files and over 1.5 seconds to the average 500KB file transfer latency. Mean packet delay and loss characteristics under light network load as measured by both `Poip` and `tcpdump` can be seen in Table 1. Delay and loss measurements are for the outbound path from the server to the client. These network measurements show that the network conditions were fairly stable throughout the measurement period.

Under light network load and light server load, the file transfer latency had very low variability for all three file sizes. Under heavy server load, the variability increases slightly. The principal effect of server load when network load is low was to increase the average file transfer time by a relatively constant amount for each file size. These effects can be see in the cumulative distribution function (CDF) diagram for the 20KB file in Figure 2. This diagram also shows that for a small percentage of transfers, latency can increase over 6 times when server load is high.

Another way to analyze file transfers and the effects of varying server load for the lightly loaded network measurements is through the use of TCP time line plots. The time lines for typical transfers of the 20KB file can be seen in Figure 3. In these plots (as well as the rest of the time line plots in this section), the client is on the left hand side and the server is on the right. The time line plots highlight the delay incurred for all file transfers when the server is busy. After the TCP connection set

Table 1: File transfer latency statistics under light network load

File Sizes	Server Load	Files Trnsfd	Mean Latency	Std. Dev. Latency	tcpdump % pkt loss	tcpdump mean delay	Poip % pkt loss	Poip mean delay
1KB	low	4696	0.276	0.058	0.0	0.046	0.0	0.044
20KB	low	1521	0.678	0.273	0.1	0.050	0.1	0.045
500KB	low	365	2.552	0.283	0.0	0.056	0.1	0.047
1KB	high	768	1.256	1.171	0.1	0.045	0.1	0.043
20KB	high	572	1.662	1.239	0.1	0.049	0.1	0.044
500KB	high	223	4.125	1.584	0.0	0.062	0.1	0.045

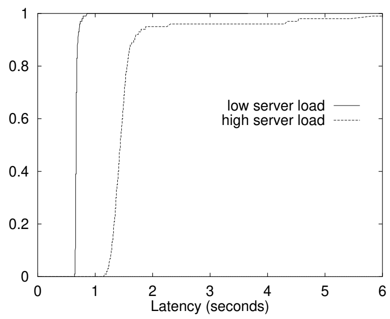


Figure 2: CDF for file transfer latency under light network load for the 20KB file

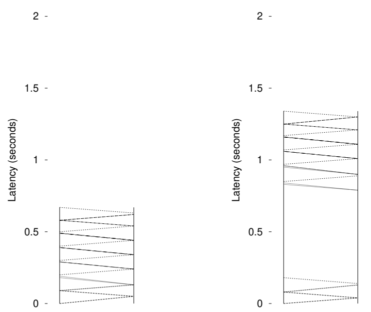


Figure 3: Time line diagram of typical 20KB file transfers under light network load

up sequence, there is roughly a one second delay before the first data packet is sent by the server. This delay is due to the need for the request from the client to get up to user space and then for the response to get back down to the network. As can be seen in the lightly loaded case, this is nearly instantaneous when the server is lightly loaded. This data combined with the information from the CDF of file transfer latency leads us to conclude that if the network is lightly loaded, the maximum additional delay that most transfers might experience when accessing our busy server is about one second.

4.1.2 Performance under heavy network load

When packet delays and losses increase, transfer characteristics change. The details of the file transfer latency under heavy network load for all three files can be seen in Table 2. The network measurements from `Poip` indicate that network conditions were fairly stable throughout the measurement period. However, these measurements do not correlate closely with the mean delay and loss values extracted from the `tcpdump` traces. We explore this in greater detail in Section 4.2. Packet transfer delays extracted from the `tcpdump` traces increase between 13% and 48% from light network load to heavy network load. The increase in packet delays is an indication of the level of congestion in the network. The table shows that under light server loads, the mean transfer latency increases in a range from 2.2 to 5 times the mean latency when the network is lightly loaded. It also shows that under heavy server loads, the mean transfer latency increases in a range from 1.3 to 2 times the mean latency when the network is lightly loaded. Perhaps the most interesting statistic to note is that transfer latency of the 500KB file under heavy network loads actually *decreases* when the server load is high versus when it is low. This effect is explored in more detail in Section 4.1.4.

File transfer delays are strongly influenced by both congestion and packet loss. Measurements taken during the day time periods for all data sets showed higher packet loss and delay characteristics as would be expected. Under heavy network load,

Table 2: File transfer latency statistics under heavy network load

File Sizes	Server Load	Files Trnsfd	Mean Latency	Std. Dev. Latency	tcpdump % pkt loss	tcpdump mean delay	Poip % pkt loss	Poip mean delay
1KB	low	1543	0.728	1.373	4.2	0.052	5.0	0.051
20KB	low	656	1.503	1.827	3.8	0.056	5.9	0.051
500KB	low	70	12.977	4.501	3.2	0.083	6.4	0.053
1KB	high	610	1.598	1.329	2.7	0.051	6.4	0.052
20KB	high	444	2.127	2.169	1.9	0.056	4.3	0.050
500KB	high	110	8.297	2.993	1.5	0.085	5.8	0.051

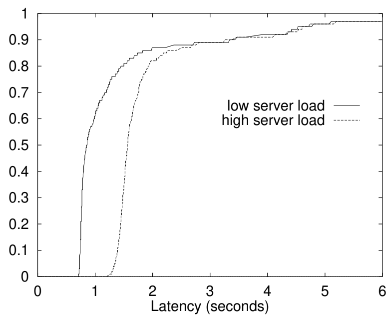


Figure 4: CDF for file transfer latency under heavy network load for the 20KB file

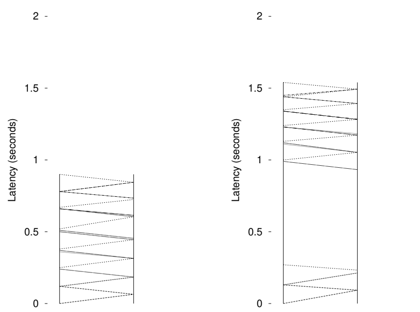


Figure 5: Time line diagram of typical 20KB file transfers under heavy network load

file transfer latency had generally higher delay and higher variability for all three file sizes regardless of server load. These effects can be seen by comparing Figure 2 with the corresponding figure for heavy network load (Figure 4). While in Figure 2, the knee in the CDF occured around the 90^{th} percentile, in Figure 4, the knee is much lower indicating that a much larger fraction of transfers experienced severe delays.

The time lines for typical transfers of the 20KB file when the network is heavily loaded can be seen in Figure 5. As in the lightly loaded network case, the delay in sending the first data packet from the server is evident. These diagrams show a general elongation in packet transfer times versus the lightly loaded network time lines, which is an indication of the level of congestion in the network during the measurement.

4.1.3 Effects of Packet Loss

Measurements made during high network load show the effects of both network congestion and packet loss. In this section we specifically examine the effects of packet loss on file transfer latency. The details of the effects of packet loss under heavy network load and high server load can be seen in Table 3. This table shows the differences between files transferred with and without packet loss. File transfers for files which lose at least one packet are between 1.3 and 7.8 times slower than files transferred without packet loss. As might be expected, variability for files transferred without loss is also much lower than that of files transferred with loss.

Figure 6 compares the impact of packet loss on file transfer latency for the 1KB file and the 20KB file. As can be seen from the figure, packet loss can significantly increase both mean and variability of file transfer delay. When comparing this figure with Figure 4 it can be seen that almost all long transfer delays are due to packet drops. One notable feature of the figure is the cusp at about the 40% level for the 1KB file. This is due to lost SYN packets. There is a three second timeout period for the re-sending of a lost SYN packet in the Linux TCP implementation.

The time lines for transfers of a typical 20KB file with and without packet loss can be seen in

Table 3: File transfer statistics under heavy network load for files with no dropped packets or at least one dropped packet for 500KB file

File Sizes	Server Load	Files w/o pkt loss	Mean Latency	Std. Dev. Latency	Files with pkt loss	Mean Latency	Std. Dev. Latency
1KB	low	1305	0.326	0.062	238	2.936	2.541
20KB	low	351	0.784	0.066	305	2.331	2.428
500KB	low	0	n/a	n/a	70	12.976	4.501
1KB	high	536	1.337	0.763	74	3.491	2.771
20KB	high	314	1.869	1.956	130	2.752	2.511
500KB	high	6	3.947	1.053	104	8.548	2.548

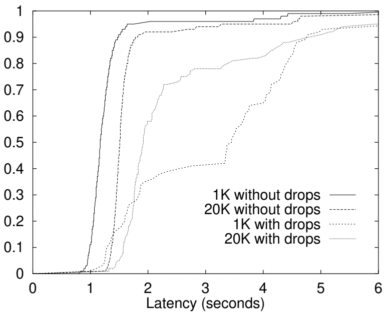
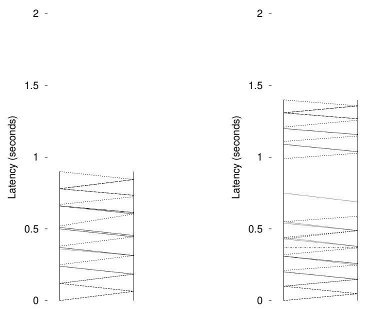


Figure 6: CDF for file transfer latency under heavy network load and heavy server load for files with and without packet loss



without packet loss with packet loss

Figure 7: Time line diagram of a 20KB file transfers under heavy network load and low server load

Figure 7. This figure shows the effect of a single lost packet during the data transfer phase of the HTTP transaction under light server load and heavy network load. The packet loss will cause TCP to restart slow start in order to transfer the remaining packets. The overall effect of this loss is the addition of about 0.5 seconds to the file transfer latency.

4.1.4 Effects of server load

Initial effects of server load on transfer latencies have been shown in the preceding sections. However, closer examination of the data from all ten data sets shows an interesting phenomenon. For transfers of the 500KB file, in seven out of ten of the data sets, packet loss actually goes *down* when server load is *increased* during the tests. Furthermore, in three out of those seven cases, file latency *decreases* when server load is *increased*. This effect can be seen in Figure 8 for one of these cases which is the same as the data in Table 2. This result can possibly be attributed to one of two things. First, the experiments for the 500KB file transfer are run approximately 45 minutes apart and the network activity could change enough during this time to cause performance to improve. However, active measurements indicate that the network was fairly consistent across all tests. Another possible explanation is that the heavy load on the server prevents it from producing as many bursts of packets as it might normally if it were lightly loaded. We intend to further investigate this effect under a wider range of conditions and conclusively determine its cause.

4.2 Active versus passive measurements of network conditions

As was seen in the previous section, values for delay and loss from `Poip` measurements and from the TCP streams themselves often disagree. This difference is important because it suggests that measurements derived from `Poip` may not be indicative of the expected performance of TCP connections. In this section we explore these differences

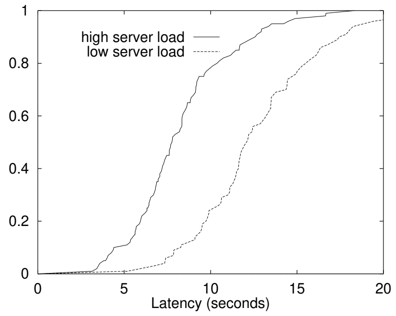


Figure 8: File transfer latency for the 500KB file transfers under heavy network load

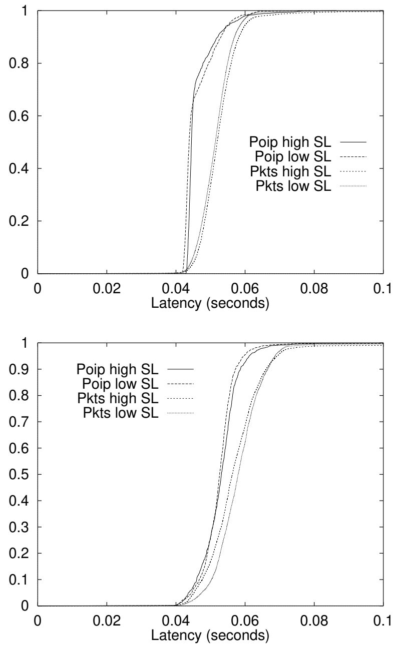


Figure 9: CDF of packet delay from server to client during two 500KB file tests (top: lightly loaded network; bottom: heavily loaded network; SL = server load).

and their implications.

Our first observation is that Poip's measurements of packet delay are typically lower than those taken from TCP streams. In Figure 9 we show the CDFs of packet delay experienced by two sets of 500KB TCP flows and by Poip during the same two time intervals. The plot on the left corresponds to lightly loaded network conditions; the plot on the right corresponds to heavily loaded network conditions. Typical differences between the mean values reported by Poip and those experienced by TCP packets are on the order of 10 ms. Note that the Poip measurements were made concurrently with the TCP connections, so they are measuring the same network state.

The higher delay experienced by TCP packets compared to packets arriving via a Poisson process is consistent with the notion that TCP's packet stream is burstier (*i.e.,* interarrivals have a higher coefficient of variation) than that of a Poisson process. This seems clear when considering that TCP often transmits packets in bursts on the order of the congestion window size, then waits for acknowledgments. The start of the next burst is determined by round trip time and generally occurs some time after the current burst has been sent, leading to a high coefficient of variation in interpacket spacing.

Our second observation is that packet loss as experienced by TCP and by Poip are not strongly related. Figure 10 shows a scatter plot of loss rate experienced by TCP (on the x axis) versus that experienced by Poip during the same time frame (on the y axis) for all of our 500 KB experiments. If the two measurements were strongly related, points would tend to occur around the line $y = x$ (which is plotted for reference); they do not. In general, it seems that Poip sees a higher packet loss rate than TCP, although this is not universally true.

The reason for this lack of correlation may be that—unlike the case with packet delays—packet losses trigger feedback behavior in TCP. Upon encountering packet loss, TCP often enters a state in which no packets flow for some time, after which it recovers and packets begin to flow again at a reduced rate (for example, see Figure 7, right side). This behavior means that TCP's view of the network is not independent of packet loss events; as a result, it may be that TCP tends to observe fewer loss events than are seen by a Poisson process (which is independent of packet loss events). It seems clear in any case that TCP's feedback behavior means that its view of network losses are not strongly correlated with that of a time-average view.

These two observations have implications for the use of Poip and similar tools to measure network state. It seems that Poip has only limited value in predicting the nature of network conditions as experienced by TCP.[1] As regards packet delays, the bursty nature of TCP flows means that packets typically see larger queues than the time-

[1] To be fair, it is important to note that Poip was not designed to assess TCP's view of the network, and that it has many uses other than the one that we are considering.

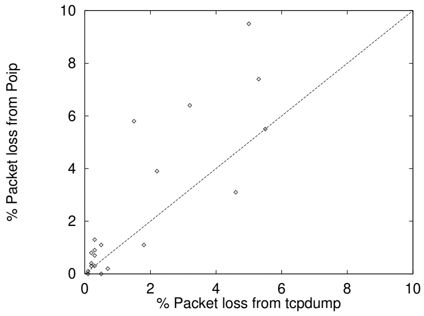


Figure 10: Scatter plot of packet loss rate in TCP vs. Poip.

averaged conditions. With respect to packet loss, it seems that the feedback nature of TCP's congestion avoidance makes it very hard to predict TCP's actual packet losses using an open-loop tool like `Poip`.

5 Conclusion

In this paper we have described the WAWM project and argued that it represents an approach that is both novel and useful. Its novelty arises from its treatment of the server and network as an integrated system. Its utility comes from the ability to explore the interaction between server behavior and network behavior, and to identify cases in which the two do not interoperate well.

We have described a small-scale implementation of the project architecture and shown that it can yield informative measurements. For example, using it we have shown that (for our server) the main effect of server load on typical transfers is to delay the *first* data packet sent. In addition we have shown that in many of our experiments, servers under high load suffered significantly *less* packet loss than those under low load. In comparing packet delay and losses in TCP connections to measurements obtained with Poip we find that there are considerable (and understandable) differences. These results are only initial looks at our data and require confirmation in a wider range of settings; however they are suggestive of interesting effects.

Our motivating question in this project is: Why is the Web so slow? To use the WAWM infrastructure to address this question will require progress along two dimensions. First, we need to expand the measurement apparatus until it allows us to assess *representative* behavior of the Web. This means that we need to add more remote clients (from more locations), and consider a wider range of server and platforms (*e.g.,* IIS running on Windows NT).

The second dimension is that of analytic methods. We intend to develop characterization methods that operate on traces of individual transfers and assess the relative impact of server delay, network delay, packet loss, etc. on transfer latency. Using these more detailed analyses we hope to obtain a more precise understand of the causes of transfer latency in the Web.

Acknowledgements. The authors would like to thank David Martin from the University of Denver for access to the system which was used as the remote client for the tests presented in this paper. The authors would also like to thank Vern Paxson for providing Poisson Ping for use in this study as well as providing input on the project design. The authors would also like to thank Lars Kellogg-Stedman for his help in setting up systems which were used in the project.

References

[1] A. Adams, J. Mahdavi, M. Mathis, and V. Paxson. Creating a scalable architecture for internet measurement. *http : //www.psc.edu/ ∼ mahdavi/nimi_paper/NIMI.html*, 1998.

[2] J. Almeida, V. Almeida, and D. Yates. Measuring the behavior of a world wide web server. In *Proceedings of the Seventh IFIP Conference on High Performance Networking (HPN)*, White Plains, NY, April 1997.

[3] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira. Characterizing reference locality in the WWW. In *Proceedings of 1996 International Conference on Parallel and Distributed Information Systems (PDIS '96)*, pages 92–103, December 1996.

[4] M. Arlitt and C. Williamson. Internet web servers: workload characterization and performance implications. *IEEE/ACM Transactions on Networking*, 5(5):631–645, October 1997.

[5] H. Balakrishnan, V. Padmanabhan, S. Seshan, M. Stemm, and R. Katz. Tcp behavior of a busy internet server: Analysis and improvements. In *Proceedings of IEEE INFOCOM '98*, San Francisco, CA, March 1998.

[6] P. Barford and M. Crovella. Generating representative workloads for network and server performance evaluation. In *Proceedings of ACM SIGMETRICS '98*, pages 151–160, Madison, WI, June 1998.

[7] P. Barford and M. Crovella. A performance evaluation of hyper text transfer protocols. In *To appear in Proceedings of ACM SIGMETRICS '99*, Atlanta, GA, May 1999.

[8] J. Bolot. End-to-end packet delay and loss behavior in the internet. In *Proceedings of ACM SIGCOMM '93*, San Francisco, Setpember 1993.

[9] L. Brakmo, S. O'Malley, and L. Peterson. Tcp vegas: New techniques for congestion detection and avoidance. In *Proceedings of ACM SIGMETRICS '96*, Philadelphia, PA, May 1996.

[10] R. Carter and M. Crovella. Measuring bottleneck link speed in packet-switched networks. In *Proceedings of Performance '96*, Lausanne, Switzerland, October 1996.

[11] National Laboratory for Applied Network Research. http://www.nlanr.net, 1998.

[12] Cooperative Association for Internet Data Analysis. http://www.caida.org, 1998.

[13] H. Frystyk-Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. Wium-Lie, and C. Lilley. Network performance effects of HTTP/1.1, CSS1 and PNG. In *Proceedings of ACM SIGCOMM '97*, Cannes, France, Setpember 1997.

[14] S. Gribble and E. Brewer. System design issues for internet middleware services: Deductions from a large client trace. In *Proceedings of the 1997 USENIX Symposium on Internet Technologies and Systems (USITS '97)*, Monterey, CA, December 1997.

[15] Keynote Systems Inc. http://www.keynote.com, 1998.

[16] INSoft. http://www.vitalsigns.com, 1998.

[17] V. Jacobson. traceroute. ftp://ftp.ee.lbl.gov/traceroute.tar.Z, 1989.

[18] V. Jacobson. pathchar. ftp://ftp.ee.lbl.gov/pathchar/msri-talk.ps.gz, 1997.

[19] S. Khaunte and J. Limb. Statistical characterization of a world wide web browsing session. Technical Report GIT-CC-97-17, College of Computing, Georgia Institute of Technology, 1997.

[20] K. Lai and M. Baker. Measuring bandwidth. In *Proceedings of IEEE INFOCOM '99*, New York, NY, March 1999.

[21] B. Mah. An empirical model of HTTP network traffic. In *Proceedings of INFOCOM '97*, Kobe, Japan, April 1997.

[22] S. Manley and M. Seltzer. Web facts and fantasy. In *Proceedings of the 1997 USENIX Symposium on Internet Technologies and Systems*, Monterey, CA, December 1997.

[23] M. Mathis. Empirical bulk transfer capacity. ftp://ftp.advanced.org/pub/IPPM/treno.txt, July 1997.

[24] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *Computer Communications Review*, 27(3), July 1997.

[25] Internet Performance Measurement and Analysis Projecct. http://nic.merit.edu/ipma/, 1998.

[26] Internet Protocol Performance Metrics. http://www.advanced.org/ippm/index.html, 1998.

[27] D. Mills. Network time protocol (version 3): Specification, implementation and analysis. Technical Report RFC 1305, Network Information Center, SRI International, Menlo Park, CA, 1992.

[28] D. Mills. Improved algorithms for synchronizing computer network clocks. *IEEE/ACM Transactions on Networking*, 3(3):245–254, June 1998.

[29] J. Mogul. The case for persistent-connection HTTP. Technical Report WRL 95/4, DEC Western Research Laboratory, Palo Alto, CA, 1995.

[30] J. Mogul. Network behavior of a busy web server and its clients. Technical Report WRL 95/5, DEC Western Research Laboratory, Palo Alto, CA, 1995.

[31] J. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy. Potential benefits of delta encoding and data compression for HTTP. In *Proceedings of ACM SIGCOMM '97*, Cannes, France, Setpember 1997.

[32] S. Moon, P. Skelley, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *Proceedings of IEEE INFOCOM '99*, New York, NY, March 1999.

[33] Coral: Passive network traffic monitoring and statistics collection. $http://www.caida.org/tools/coral$.

[34] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM '98*, Vancouver, Canada, Setpember 1998.

[35] V. Paxon. Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic. *Computer Communications Review*, 27(5):5–18, October 1997.

[36] V. Paxson. Empirically-derived analytic models of wide-area tcp connections. *IEEE/ACM Transactions on Networking*, pages 316–336, August 1994.

[37] V. Paxson. End-to-end routing behavior in the internet. In *Proceedings of ACM SIGCOMM '96*, Palo Alto, CA, August 1996.

[38] V. Paxson. Automated packet trace analysis of tcp implementations. In *Proceedings of ACM SIGCOMM '97*, Cannes, France, September 1997.

[39] V. Paxson. End-to-end internet packet dynamics. In *Proceedings of ACM SIGCOMM '97*, Cannes, France, September 1997.

[40] V. Paxson. On calibrating measurements of packet transit times. In *Proceedings of ACM SIGMETRICS '98*, pages 11–21, Madison, WI, June 1998.

[41] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. Framework for IP performance metrics, RFC 2330. ftp://ftp.isi.edu/in-notes/rfc2330.txt, 1998.

[42] Apache HTTP Server Project. http://www.apache.org, 1998.

[43] The Surveyor Project. $http://www.advanced.org/csg-ippm/$, 1998.

[44] SSH Communications Security. $http://www.ssh.fi$, 1999.

[45] CAIDA Measurement Tool Taxonomy. $http://www.caida.org/tools/taxonomy$.

[46] tcpdump. $http://ftp.ee.lbl.gov/tcpdump.tar.Z$.

[47] G. Trent and M. Sake. Webstone: The first generation in http server benchmarking, February 1995. Silicon Graphics White Paper.

[48] M. Yajnik, S. Moon, J. Kurose, and D. Towsley. Measurement and modeling of temporal dependence in packet loss. In *Proceedings of IEEE INFOCOM '99*, New York, NY, March 1999.