Performance Characteristics of Mirror Servers on the Internet

Andy Myers and Peter Dinda and Hui Zhang

July 15, 1998

CMU-CS-98-157

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

As a growing number of web sites introduce mirrors to increase throughput, the challenge for clients becomes choosing which mirror will offer the best performance when a document is to be retrieved. In this paper we present findings from measuring 9 clients scattered throughout the United States retrieving over 490,000 documents from 45 production web servers which mirror three different sites. Our findings are intended as an aid in the design of protocols for choosing among mirrored servers. Though server performance varies widely, we have observed only small, time independent variations in the relative performance of mirror servers. Further, a change in an individual server's transfer time is not a strong indicator that its performance relative to other servers has changed. We have also found that clients wishing to achieve near-optimal performance may only need to consider a small number of servers, rather than all mirrors of a particular site. Finally, we noticed that the choice of document affects the choice of server in a significant fraction of the fetches performed.

This research was sponsored by DARPA under contract numbers N66001-96-C-8528 and E30602-97-2-0287, and by a NSF Career Award under grant number NCR-9624979. Additional support was provided by Intel Corp., MCI, and Sun Microsystems. Views and conclusions contained in this document are those of the authors and should no be interpreted as representing the official policies, either expressed or implied, of DARPA, NSF, Intel, MCI, Sun, or the U.S. government.

Keywords: WWW, web server, mirroring, measurement

1 Introduction

Replicating servers in a distributed manner on the Internet has been employed for many years as a way to increase reliability and performance in the presence of frequent accesses by many clients. The weakness of replication, or mirroring, is that clients are not given a standard mechanism for selecting among the mirrors. More recently, Partridge et al. [18] have proposed *anycast*, a scheme that allows a client to automatically reach the closest replica of a server. Others [9, 4] have observed that static metrics of proximity, such as distance in hops, are less effective at finding a server that will deliver good performance than metrics which take dynamically changing network and server conditions into account.

This paper presents results of a study based on client probing of web servers that we have undertaken to gain insight on approaches to designing anycast protocols. Our focus is on characterizing the performance a client receives when transferring documents from servers. We wish to answer two questions:

- How difficult is it for a client to achieve good performance?
- What are effective methods for a client to use in finding a server that will offer good performance?

To answer the first question, we have looked at the variation of performance across servers and the variation of performance at a single server. If the client observes good performance from all, or many, servers, or if a single server provides consistently good performance, the construction of an anycast protocol will be significantly easier.

To answer the second question, we have looked at possible signs that a server's performance is going to change. We have explored the dependence of a server's rank (among other servers) on time to determine how quickly server performance data ages. We have also examined the relationship of changes in a server's performance to changes in a server's rank. This will help determine whether a client can judge a server based on increases or decreases in its performance over time.

We have four main results:

- The probability of any server's rank change depends very little on the time scale over which the rank change takes place.
- There is a weak but detectable link between a server's change in transfer time and its change in rank.
- Clients can achieve near-optimal performance by considering only a few servers out of the whole group of mirrors.
- Server choice depends on document choice.

We discuss the implications of these results in Section 9.

There is a long tradition of Internet measurement studies [1, 2, 5, 6, 8, 10, 11, 12, 14, 15, 16, 17, 19, 20, 21, 22, 24]. While some of this work, notably Balakrishnan et al.'s [2], has been focused on the World Wide Web, we are only aware of a few limited studies that look at servers as a group and measure their relative performance. Bhattarcharjee et al. [3] measured "server response time,"

Client Site	Avg. time of one group	Number of fetches	Failure rate
Carnegie Mellon	0:32:49 hours	54695	10.18%
Georgia Tech.	0:23:47	60021	11.55%
ISI	0:36:31	53200	22.13%
U. of Califorinia, Berkeley	0:32:33	55062	4.62%
U. of Kentucky	0:31:14	55091	12.76%
U. of Mass., Amherst	1:10:34	36542	10.95%
U. of Texas	0:39:34	51640	4.70%
U. of Virginia	0:19:19	62405	28.88%
Washington U., St. Louis	0:23:16	62187	1.96%

Figure 1: Average time for one round of fetches, number of fetches completed, and failure rate for each client site

defined to be the time required to send a query to a server and receive a brief response, using clients at a single site to visit two sets of web sites. While neither set of sites were true mirrors, each set consisted of servers with similar content. Bhattacharjee also measured the throughput between a client and four FTP servers. Also, Carter and Crovella [9] measured ping times and hop counts to 5262 web servers to measure how well one approximated the other. In contrast, our study is on a larger scale, using multiple client sites, a longer measurement period, and a larger number of groups of popular web servers that are true mirrors.

The rest of this paper will consist of a description of our data collection system (Section 2), a general picture of the data we collected (Sections 3 and 4), a discussion of our findings (Section 5 through Section 8), implications of our results (Section 9), and conclusions (Section 10).

2 Data collection methodology

At each of nine client sites where we had guest accounts (listed in Figure 1) we fetched documents from each server in three sets of mirrored web sites (the Apache Web Server site, NASA's Mars site, and News Headlines). The Apache and Mars web sites were true mirrors: each of the servers in one set held the same documents at the same time. However, the News sites were an artificial mirror since they did not contain the same documents. The News servers were picked from Yahoo's index (http://www.yahoo.com/). Current headlines from each of the News sites were fetched and the transfer times were normalized so that all News documents appeared to be 20 KB long. For the mars and Apache servers, we used five documents ranging in size from 2 KB to 1.3 MB (listed in Figure 3).

Clients visited servers sequentially, fetching all documents from a server before moving on to the next. Similarly, all mirrors of one site were visited before moving on to the next. For example, a client would start by visiting http://www.sgi.com/, the first Mars mirror on the list, and fetching each of the Mars documents from it. Then the client would fetch the Mars documents from the second Mars server, then the third, and so on. When all of the Mars servers had been visited, the client would move on to the Apache mirrors, and finally to the News sites. We will refer to the process of visiting all servers and collecting all documents once as a *group* of fetches.

Mars sites	
http://mars.sgi.com	http://www.sun.com/mars
http://entertainment.digital.com/mars/JPL	http://mars.novell.com
http://mars.primehost.com	http://mars.hp.com
http://mars.excite.com/mars	http://mars1.demonet.com
http://mars.wisewire.com	http://mars.ihighway.net
http://pathfinder.keyway.net/pathfinder	http://mpfwww.arc.nasa.gov
http://mars.jpl.nasa.gov	http://www.ncsa.uiuc.edu/mars
http://mars.sdsc.edu	http://laguerre.psc.edu/Mars
http://www.ksc.nasa.gov/mars	http://mars.nlanr.net
http://mars.catlin.edu	http://mars.pgd.hawaii.edu
News sites	
http://www.cnn.com/	http://www.nytimes.com/index.gif
http://www.latimes.com/	http://www.washingtonpost.com/
http://www.csmonitor.com/	http://www.usatoday.com/
http://www.abcnews.com/	http://www.msnbc.com/
http://www.s-t.com/	http://nt.excite.com/
http://news.bbc.co.uk/	http://www.newscurrent.com/
http://pathfinder.com/time/daily	http://www.sfgate.com/news/
http://headlines.yahoo.com/Full_Coverage/	http://www.topnews.com/
Apache sites	
http://www.rge.com/pub/infosystems/apache/	http://apache.compuex.com/
http://apache.arctic.org/	http://ftp.epix.net/apache/
http://apache.iquest.net/	http://www.apache.org/
http://apache.utw.com/	http://www.ameth.org/apache/
http://apache.technomancer.com/	http://apache.plinet.com/
http://fanying.eecs.stevens-tech.edu/pub/mirrors/apache/	

Figure 2: Servers visited

After all servers were visited, the client would sleep for a random amount of time taken from an exponential distribution with a mean of $1/2$ hour added to a constant $1/2$ hour. By scheduling the next group of fetches relative to the previous group's finish time (rather than its start time), we avoided situations in which multiple fetches from the same client interfered with each other, competing for bandwidth on links near the client.

We introduced the delay between fetches to limit the impact of our measurements on client and server sites. A typical group of fetches involved transferring more than 60 MBytes of data to a client. If the fetches finished in 30 minutes, the average transfer rate would have been 266 Kbps, which is a noticeable share of the traffic on a LAN. The delay between groups of fetches lowered the average resource utilization to a more acceptable level. Another difficulty with more frequent fetches was an increase in the likelihood that our own fetches would have affected the quantity (throughput) we were trying to measure.

	URL	Size (bytes)
Mars documents		
0	/nav.html	2967
1	/2001/lander.jpg	70503
2	/mgs/msss/camera/images/12_31_97_release/2303/2303p.jpg	235982
3	/mgs/msss/camera/images/12_31_97_release/2201/2201p.jpg	403973
4	/mgs/msss/camera/images/12_31_97_release/3104/3104p.jpg	1174839
Apache documents		
0	dist/patches/apply_to_1.2.4/no2slash-loop-fix.patch	1268
1	dist/CHANGES_1.2	90631
2	dist/contrib/modules/mod_conv.0.2.tar.gz	74192
3	dist/apache_1.2.6.tar.gz	714976
4	dist/binaries/linux_2.x/apache_1.2.4-i586-whatever-linux2.tar.Z	1299105

Figure 3: URLs of documents fetched from Mars and Apache servers

We used the lynx[1] web browser to perform fetches. Choosing lynx was a compromise between realism and ease of implementation. Lynx is an actual production web browser that people use every day. At the same time, it is easy to control via command line switches, allowing us to run fetches via a perl script. Implementing our own URL fetch code might not have captured the characteristics of actual browsers. Conversely, using a more popular, hence more realistic, browser, e.g. Netscape, would have presented a significant programming challenge.

Our client script would invoke lynx to retrieve a URL and send it to standard output. The number of bytes received by lynx was counted and recorded along with the amount of time the fetch took to complete. If a fetch did not terminate after five minutes, it would be considered unsuccessful and the associated lynx process would be terminated. We chose five minutes as a compromise between achieving a complete picture of a server's behavior and forcing groups of fetches to complete in a reasonable amount of time. The observable effects of such a short timeout are a slightly higher failure rate, especially among larger documents. Other possible causes for timeouts are network partitions, client errors (lynx might have frozen), server errors (the server might have stopped providing data), or major shortages of available bandwidth. In our analysis, we treat these incidents as failures to collect data, rather than as failures of servers.

Fetches could also be unsuccessful if the number of bytes returned was incorrect. We found that the wrong number of bytes usually indicated a temporary failure such as a "server too busy" message although in some cases it signified that the server no longer existed (failed DNS query) or was no longer mirroring data. We assumed that every fetch which returned the proper number of bytes succeeded.

It was more difficult to identify failed fetches from the News sites. Since we were retrieving news headlines, each page's content was constantly changing, meaning we could not use a hard-coded size to determine success. A simple heuristic that worked well was to assume that all fetches that returned less than 600 bytes were failures. This value was larger than typical error messages (200-300 bytes) and smaller than typical page sizes (as low as 3k on some servers). As with the

[1] Available from http://lynx.browser.org/

other servers, fetches lasting five minutes were considered failures.

While our fetch scripts were running, there were multiple occassions on which client machines crashed or were rebooted. To limit the impact of these interruptions, we used the Unix `cron` system to run a "nanny" script every 10 minutes which would restart the fetch script if necessary. This kept all fetch scripts running as often as possible.

2.1 Limitations

While our methodology was sufficient to capture the information in which we were most interested, there were some data that we were not able to capture. Because of the relatively large, random gaps between fetches to the same server, we were unable to capture shorter-term periodic behavior. Further, because each group of fetches finished in a different amount of time according to server load and network congestion at the time, the distribution of interarrivals of fetches to a single server from a client was extremeley hard to characterize and exploit. Thus, we were unable to map the observed frequency of network conditions to the actual frequency of occurrence of these conditions.

No two fetches from a given client were done simultaneously to prevent the fetches from competing with each other. At the same time, we would like to compare results across servers to rank servers relative to one another. There is a reasonable amount of evidence which suggests that network performance changes over longer time scales [23][2] while our measurements took place over shorter time scales. On average, clients visited all Mars mirrors in just over 17 minutes, all Apache mirrors in under 13 minutes, and all News sites in less than one and a half minutes. Because of these results, we believe that it is valid to treat sequential fetches as occurring simultaneously.

Another artifact of sequential fetches is that periods of network congestion are possibly underrepresented in the data. As congestion increases, fetches will take longer. The result is that the number of fetches completed during periods of congestion will be lower than the number completed during periods with less congestion. If periods of congestion are short-lived, only a few fetches will reflect the congestion. If periods of congestion are long-lived, all fetches will take longer but the total number of groups of fetches completed will be smaller.

Finally, we must consider inter-client effects. Because each client's fetches are independently scheduled, two clients could wind up visiting the same server at the same time. We will refer to such an incident as a *collision*. We believe that collisions have a negligible effect on fetch times. Further, less than 10% of all fetches were involved in collisions.

3 Data Characteristics

All clients began fetching documents on the afternoon of Thursday, April 23, 1998 and continued until the morning of Thursday, May 14, 1998. During this 3 week period, there were a total of 490843 fetches made. By data set, there were 287209 fetches to Mars servers, 157762 to Apache servers, and 45872 to News servers. The much lower number for the News data is mostly due to the fact that we only fetched one document from each News site compared to five from each Mars and Apache site. We can estimate the number of times each set of servers was visited by dividing the number of fetches by the number of combinations of servers and documents. For Mars, we divide

287209 by 100 (20 servers x 5 documents) to find that the Mars servers were visited 2872 times. Similarly, we see that Apache servers were visited 2868 times and News servers were visited 2867 times.

The slightly lower number of visits to Apache and News sites is a product of the way the client fetch script reacted to crashes. When a client was restarted, it began fetching documents from the first server on its list rather than starting at the place where the last series of fetches left off. Since clients visited Mars sites first, then Apache sites, and finally News sites, it is not surprising that there are more fetches to Mars sites than to Apache sites and more fetches to Apache sites than to News sites.

The number of fetches performed and the average length of time that one group of fetches took to complete at each client site can be found in Figure 1. As expected, sites with longer group fetch times completed fewer fetches. We believe the differences across clients reflect variation in the amount of available bandwidth and machine speed at each client site.

Figure 1 also shows the percentage of fetches that reached the five minute timeout and were classified as failures. By client, the proportion of failures ranged from 1.96% to 22.13% of fetches. Considering the loss rate by server set, we see that Mars servers failed 5.85% of the time, News servers failed 9.49% of the time, and Apache servers failed 24.23% of the time. As far as we can tell, the differences in failure rates across types of mirrors are not the result of using one brand of web server or another. However, we did notice that three Apache servers consistently timed out for some clients while they succeeded a reasonable amount of time for other clients. These three servers account for most of the Apache servers' comparatively higher failure rate.

3.1 Ranks

Throughout this paper, we will use *rank* to compare servers' performance. Here we will explain how rankings are computed and give some insight into what differences in rank mean. A ranking of servers is computed for each data set (Mars, News, or Apache) for each group of fetches at each client. Recall that after each group of fetches, a client has performance data for each web server. For each document, we can order the servers by their fetch times from lowest to highest, discarding those servers whose fetches failed. A server's rank is merely its place in this order. The server which comes first in the order has the highest rank (0), the server which comes next has a rank of 1, and so on. In our terminology, lower ranks correspond to better server performance. In summary, each successful group of fetches generates one set of ranks for each of the 11 documents: 5 sets for Mars documents, 5 for Apache documents, and one for the News document.

There is some inaccuracy in our method of ranking servers: The tacit assumption in computing ranks is that the fetch times being compared were generated under identical conditions. As we have discussed in Section 2.1, this is not possible, but we believe that network conditions do not change a significant amount between the first and last fetch of a document from a group of servers.

Ranks are not significant performance indicators by themselves. Ranks will not say whether or not the difference in performance between servers is negligible. But there in the data that we collected, we have found a very strong link between noticeable differences in performance and differences in rank.

Figure 4 plots the normalized, average transfer time vs. server rank for document 4 of the Mars data set. The graph shows a definite rise in transfer time as rank increases. For example, we

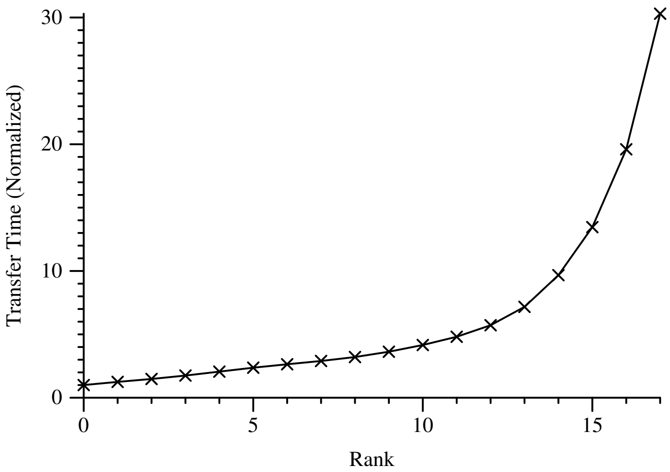


Figure 4: Average amount of separation between rank values for Mars servers, document 4, all clients aggregated.

see that on average, a server with a rank of 4 has twice the transfer time of a server with a rank of 0. Further, the server with the largest rank (17) takes more than 30 times as long to transfer a document as the best server, and it takes more than 3 times as long to deliver a document as a server with a rank of 14.

The primary point of Figure 4 is that rank changes usually correspond to noticeable performance changes for document 4 of the Mars set. All other documents from Mars, Apache, and News produced similar graphs, though the Apache and News data tended to have much larger differences in performance. This gives us confidence that ranks are a reasonable way to talk about the relative performance of servers.

4 Summary statistics and distributions

Our data consists of random samples (as we note in the next section, there is almost no significant sequential correlation in our samples) where each sample consists of a transfer time from a client to a server and its ranking relative to the other transfers in its group of fetches. This section summarizes these samples in terms of general statistics and analytic distributions. Conceptually, the analysis gives some insight into what a random client can expect from a random mirror site for different sizes and kinds of documents. There are two main results here. First, transfer times and server rankings exhibit considerable variablility. Second, transfer times, taken to a fractional power, are well fit by an exponential distribution.

The analysis is from the point of view of a random client site (from Figure 1) attempting to fetch a particular document from a set of mirror sites (Figure 2.) There are 11 different combinations here (Apache and Mars each serve five different documents while News serves one virtual document.) For each of these combinations, we examine the transfer times and corresponding ranks for all the client fetches of the document to the set of mirror sites. In effect, we factor out the set of mirrors and the document size here by doing this.

Dataset/Doc	Transfer time (seconds)					Ranks				
	Mean	StdDev	Median	Min	Max	Mean	StdDev	Median	Min	Max
Apache/0	1.9632	5.8366	.7	0.1000	230.5100	4.2790	2.9834	4	0	10
Apache/1	3.9112	7.9753	2	0.3800	297.7000	4.2737	2.9610	4	0	10
Apache/2	3.2929	6.3993	1.7	0.3000	293.9000	4.1372	2.8277	4	0	10
Apache/3	15.4776	18.2385	10.7	1.3000	299.9000	3.9916	2.7643	4	0	10
Apache/4	23.1960	22.9257	17.9	2.2000	298.2000	3.7789	2.6920	4	0	10
Mars/0	1.5416	4.6808	0.7	0.1000	296.6000	8.2060	5.0596	8	0	17
Mars/1	2.6929	6.5319	1.3	0.1000	292.6000	8.1667	5.0496	8	0	17
Mars/2	5.8062	9.4102	3.3	0.3000	290.5000	8.1287	5.0350	8	0	17
Mars/3	8.7380	12.3967	5.3	0.6000	297.3000	8.0995	5.0268	8	0	17
Mars/4	19.9019	23.5427	13.9	1.6000	298.2000	7.9213	4.9654	8	0	17
News/0	3.8185	11.8028	1.06	0.1200	638.9800	6.4049	4.0692	6	0	14

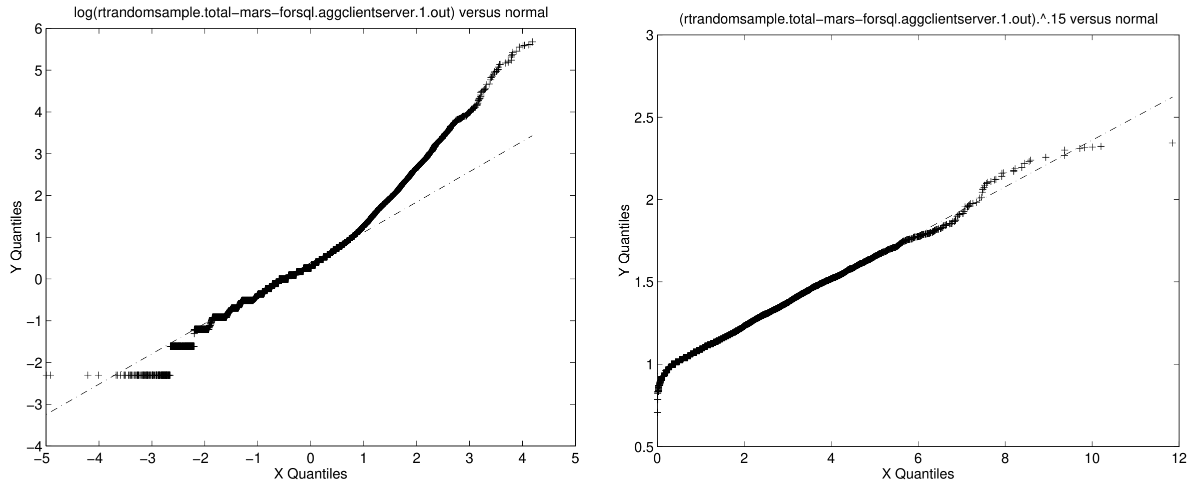
Figure 5: Summary statistics of transfer time and corresponding ranks.

Figure 5 presents the summary statistics of transfer times and ranks for each of the combinations. Notice that mean transfer times as well as standard deviations increase with increasing document size. Further, transfer times are highly variable — standard deviations are about as large as means, and we see maxima and minima near the limits we placed on observered transfer times (300 seconds.) It is important to note that the maximum transfer time of 638.98 seconds for the News/0 dataset is due to our normalizing the transfer times for News documents according to their size to approximate always fetching a 20 KB document. In some cases, particularly slow fetches can result in normalized transfer times exceeding 300 seconds. This is rare.

Figure 5 also shows statistics of ranks. An interesting observation here is that the standard deviation of ranks, although quite large, does not bode disaster for server selection algorithms. A random selection is likely to result in an average server. Further, it may well be the case that some servers vary less in their ranking than others – for example, the rankings of a few good servers may very well remain stable while the remaining servers have more drastically varying rankings. The reader may notice that the median and maximum ranks noted in the table are low - for example, there are 16 News servers, yet the maximum rank observed is 14. This effect is due to the fact that in any group there is likely to be at least one failed fetch which results in a truncation of ranks.

While summary statistics provide some insight on the performance, both absolute (transfer time) and relative (rank) a client can expect to receive, they provide a very limited view of the distribution of these quantities. To better understand the distribution of transfer times, we attempted to fit a variety of analytic distributions to the data. The quality of such a fit can be determined by a quantile-quantile plot, which plots the quantiles of the data versus the quantiles of the distribution [13] (pp. 196–200.) A good fit results in a straight line, regardless of the parameters chosen for the analytic distribution.

We tried normal, exponential, and poisson distributions. None of these fit the transfer time data very well, especially at the tails. The distribution of transfer times is heavy-tailed compared to these distributions. Next, we tried the log-normal distribution by testing if the logarithms of our data points were normally distributed. Figure 6(a) shows the quantile-quantile plot of the logarithm of the Mars/1 dataset versus normal and is representative of the other datasets. Generally, log-normal was much better than the earlier distributions. This result agrees with Balakrishnan et al.



(a) log(data) versus normal (b) data$^{0.15}$ versus exponential

Figure 6: Quantile-Quantile plots comparing distribution of transfer times in Mars/1 to (a) log-normal and (b) power-exponential.

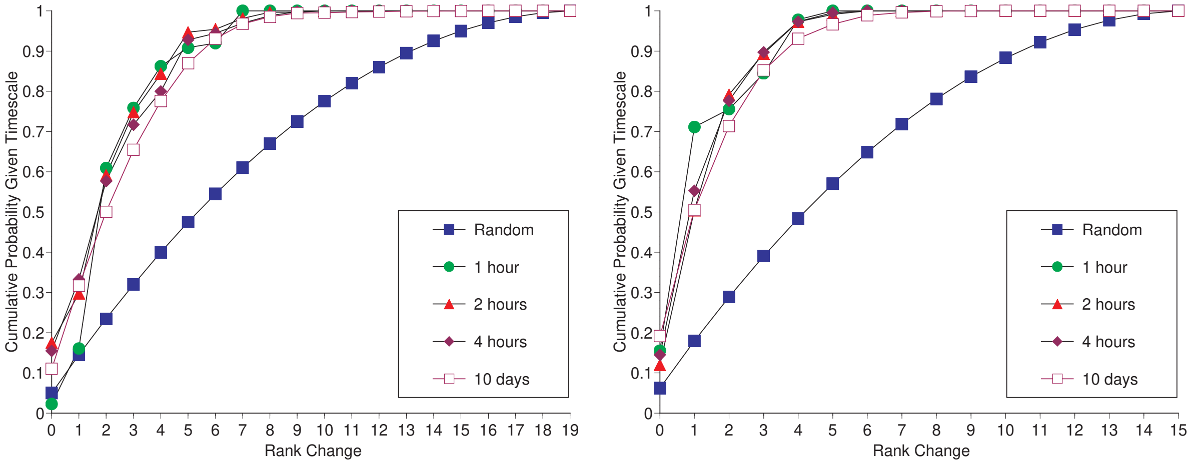
[2], who also found that a single client's observed throughput can be modeled reasonably well by a log-normal distribution.

Observing that the transformed data was a closer fit to common distributions, we next tried a power transformation — raising the data to a fractional power — and seeing if the transformed data could be fitted with a common analytic distribution. This turned out to provide the best results. For example, in Figure 6(b) we have raised the data to the 0.15 power and plotted the quantiles of the transformed data versus the quantiles of an exponential distribution. The near-perfect linearity makes it clear that this "power-exponential" distribution is a particularly good fit for Mars/1. Power-exponential also fits the other datasets exceedingly well. Some caution must be used here, however. Because transfer times were artificially truncated at 5 minutes, we do not have an accurate picture of the tail of the distribution. It may be the case that the actual distribution of server transfer times is much more heavy-tailed, meaning that the power-exponential distribution may not fit this data as well as it seems to.

5 The timescale of rank changes

Once a client has found a highly ranked server, the client is interested in how long that server is likely to maintain a high rank among the servers the client could fetch from. Fundamentally, it is the timescale over which significant rank changes occur that is important. In this section, we show that most rank changes are small, even over relatively long time scales. Good servers remain good for long periods of time. Indeed, the probability of rank changes depends very little on the time scale over which they occur.

Given periodically sampled ranks, the natural way to study change on different timescales would be via frequency domain or time series analysis [7]. However, as we discussed in Section 2, our data was collected at exponentially distributed intervals, making this difficult. The transfer time data could be resampled periodically and new rankings computed, but such resampling is



(a) Mars/1 (b) News/0

Figure 7: $P[|r_{t+w} - r_t| \leq R \mid sample\ period \leq w \leq W]$ for (a) Mars/1 and News/0 plotted for several different values of W where R is the rank change and W is the maximum time period. Other Mars plots are similar.

complex and since signal reconstruction from nonperiodic samples is an area of current research, such an approach would be questionable as well as difficult to understand. We did informally try this method and found results similar to those presented here.

Our approach was to estimate the cumulative probability of rank changes over increasing time scales. Consider a single mirror server. From the point of view of a single client using the set of mirrors, we have a sequence of time-stamped samples of that server's rank (as well as transfer times.) Now extract all the pairs of rank samples that are four or fewer hours apart. For each pair, subtract the earlier rank from the later rank and take the absolute value. Count the number of occurances of each of the possible rank changes. Accumulating these in the appropriate order gives an estimate of the cumulative probability of rank changes given measurements four or fewer hours apart.

We may find that rank changes of three or fewer are 80% probable given time scales of four or fewer hours. Notationally, we express this as $P[|r_{t+w} - r_t| \leq R \mid sample\ period \leq w \leq W] = 0.8$, where $R = 3$ is the rank change, $W = 4$ hours is the maximum time scale and the rs are our rank samples. For each combination of W and R examined, we use a randomly selected 10,000 samples to assure a tight estimate of the probability. Further, we aggregate the probabilities across all clients for each dataset and document to obtain the point of view of a random client interacting with a random server within the group. Finally, it is important to note that we are limited by our average sampling interval of one hour — we cannot discern behavior for $W < 1$ hour.

Figure 7 shows representative plots of the cumulative probability for the (a) Mars/1 and (b) News/0 datasets. The way to read these plots is to pick a time scale, follow the corresponding curve horizontally to the maximum rank change that is of interest, and then read the cumulative probability from the vertical axis. For example, considering the Mars/1 data set, we see that for time scales of two (or fewer) hours, rank changes of four (or fewer) occur with probability 0.9. The graphs also include the curve that would result if rankings were simply uniformly distributed random permutations.

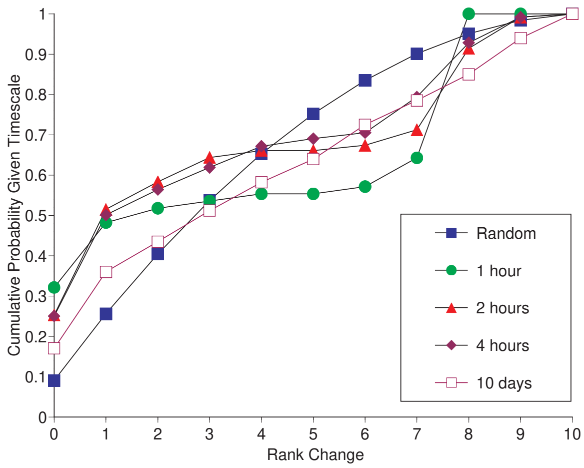


Figure 8: $P[|r_{t+w} - r_t| \leq R \mid sampleperiod \leq w \leq W]$ for Apache/1, plotted for several different values of W. Other Apache plots are roughly similar, and all Apache plots differ signicantly from Mars or News plots.

It is clear from the graphs that most rank changes are small. The 10 day curves cover the vast majority of the data, and we can see that the smallest 25% of possible rank changes account for about 90% of rank changes.

The graphs also show that rank changes depend very little on the time scales over which they occur. If there was a strong dependency, the curves for the different time scales would be more widely separated. We can see that the curves for increasingly longer time scales slowly move to the right (toward larger rank changes), but the effect is very marginal. This is a very promising result. If a client can find a good server, it is highly likely that it will remain good for quite some time.

The graphs of Figure 7 are representative for the Mars and News datasets. Unfortunately, the Apache data shows very different behavior, as can be seen in Figure 8, which shows a cumulative probability plot for a representative, Apache/1. Here, we don't see the quick rise of the curves, so large rank changes are relatively much more probable than with the Mars and News data. Further, since the curves do not hug each other very closely, there is more dependence on the time scale. At this point, we do not understand why the Apache data is so different. The clearest distinquishing characteristic of the Apache sites is that they tend to run noncommercial web servers (the Apache web server) while the Mars and News sites tend to run commercial web servers (Netscape and Microsoft servers.) We have no evidence that this difference causes the discrepency, however.

6 Changes in transfer time and rank

A client using a highly ranked server is interested in warning signs that may indicate that the server's ranking has changed dramatically. The client cannot measure rankings without measuring all of the mirror servers; it can only observe the transfer times it is experiencing on the currently chosen server. The natural question is what, if any, relationship exists between the changes in transfer time a client observes and the changes in rank the server experiences. Our study shows

Dataset/Doc	Changes in transfer time (seconds)					Changes in rank				
	Mean	StdDev	Median	Min	Max	Mean	StdDev	Median	Min	Max
Apache/5	0.0039	8.4087	0	-123.8000	226.4100	0.0091	4.2022	0	-10	10
Apache/6	-0.0810	10.3995	0	-295.7300	267.8000	0.0010	4.1948	0	-10	10
Apache/7	-0.0503	9.0000	0	-292.5000	205.9000	-0.0621	4.0177	0	-10	10
Apache/8	-0.5457	25.4940	0	-285.3000	276.5000	-0.0196	3.8818	0	-10	10
Apache/9	-0.1912	31.8086	0.1	-278.0100	287.7000	-0.0367	3.8072	0	-10	10
Mars/0	0.1068	6.0450	0	-227.9100	221.4600	0.0244	7.1711	0	-17	17
Mars/1	0.1218	8.1173	0	-184.0000	232.5900	0.1330	7.0711	0	-17	17
Mars/2	0.1189	14.3483	0	-285.2000	287.4000	-0.0685	7.1195	0	-17	17
Mars/3	-0.0226	17.5260	0	-253.6000	282.1000	-0.0038	7.0849	0	-17	17
Mars/4	0.3308	34.5870	0	-286.9000	288.6000	0.0194	7.0992	0	-17	17
News/0	0.0282	17.1793	0	-298.8300	293.8300	-0.0316	5.8363	0	-14	14

Figure 9: Summary statistics of changes in transfer time and changes in coresponding ranks.

that while a relationship does exist, it is very marginal.

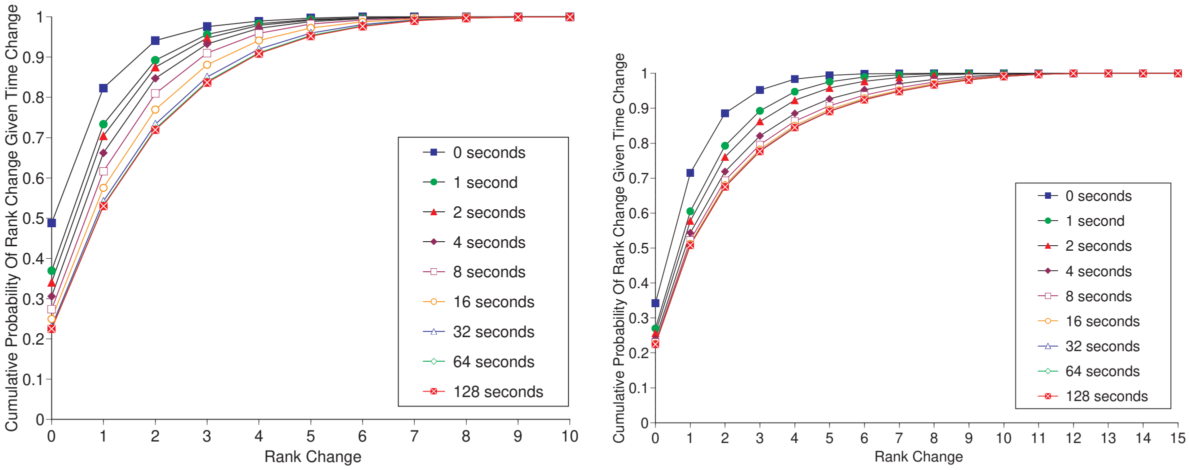
Our approach was to estimate the cumulative probability of rank changes over increasing changes in observed transfer times. Consider a single mirror server. From the point of view of a single client using the set of mirrors, we have a sequence of samples of that server's transfer times and their corresponding ranks. We form the cross product of these samples and select a random subset of 100,000 of these sample pairs. For each pair of samples in the subset, we subtract the transfer times and ranks.

Figure 9 shows the summary statistics of these changes in transfer time and corresponding rank. We see that the mean and median changes in both quantities are almost exactly zero. The distributions of these changes are also quite symetric about zero. For this reason, we concentrate on absolute changes in transfer time and rank.

After taking absolute values, we count occurances of value pairs to estimate the joint cumulative probability of absolute changes in rank and absolute changes in transfer time, $P[|r_{t_i} - r_{t_j}| \leq R \wedge |d_{t_i} - d_{t_j}| \leq D]$ where R is the rank change and D is the change in transfer time. Since changes in rank are categorical, we can then trivially compute the cumulative probability of an absolute change in rank *given* an absolute change in transfer time of D or smaller. Notationally, this is $P[|r_{t_i} - r_{t_j}| \leq R \mid |d_{t_i} - d_{t_j}| \leq D]$. We aggregate the probabilities from all clients for each dataset and document to obtain the point of view of a random client interacting with a random server within the set of mirrors. The reader may object that this scheme also aggregates changes happening at all time scales. This is true. However, recall from Section 5 that changes in rank are virtually independent of timescale.

Figure 10 shows representative plots of the cumulative probability for the (a) Apache/4 and (b) News/0 datasets. The plots for all of the datasets are similar. The way to read these plots is to pick a change in duration, follow the corresponding curve horizontally to the maximum rank change that is of interest, and then read the cumulative probability from the vertical axis. For example, considering the News/0 data set, we see that for a transfer time change of 128 seconds or less, 90% of rank changes are of seven or less.

We can see that large changes in transfer time are more likely than small changes to indicate large rank changes. The curves for increasingly larger changes in transfer time shift toward



(a) Apache/4 (b) News/0

Figure 10: Cumulative probability of rank change given changes in transfer time less than D ($P[|r_{t_i} - r_{t_j}| \leq R \mid |d_{t_i} - d_{t_j}| \leq D]$) for (a) Apache/4 and (b) News/0, plotted for several different values of D. All otehr plots are similar.

the right (toward larger rank changes.) However, the difference is slight. For example, consider Apache/4: A rank change of three or smaller is 90% probable with a change in transfer time of one second or smaller, while a change of transfer time of up to 128 seconds reduces the probability only to 80%. This is typical of the Apache data, and the relationship is even less pronounced for the other data.

Another way to see the limited relationship of changes of rank to changes in transfer time is to plot rank changes against their corresponding transfer time changes. Figure 11 shows a representative plot for the News/0 data, where we have focused on transfer time changes in the $[-10, 10]$ range. We have fit a least squares line to the data and have found that the relationship is marginal at best. The R^2 value for the line is only 0.36. For a wider range of transfer times, the fit is even worse. Clearly, there is only a limited relationship between changes in transfer time and changes in rank.

7 Small working sets

The observation in Section 5 that most rank changes are small leads us to ask how many servers must a client consider to achieve optimal performance. If server ranks never changed, a client would only need to use one server, the one with the best rank. But because server ranks do change, a client will need to evaluate multiple servers to find the current best. We have found that a client needs to evaluate a very small number of servers, usually less than half the total number of servers, to achieve near-optimal performance. In this section, we define a server's performance to be near-optimal, or "good," if it can deliver a document in no longer than 10% more than the time it takes the current best server to deliver the same document.

We define a *working set* to be the minimum subset of servers from a group of mirrors that can provide near-optimal performance to a client. If a working set contains all the mirrors of a site,

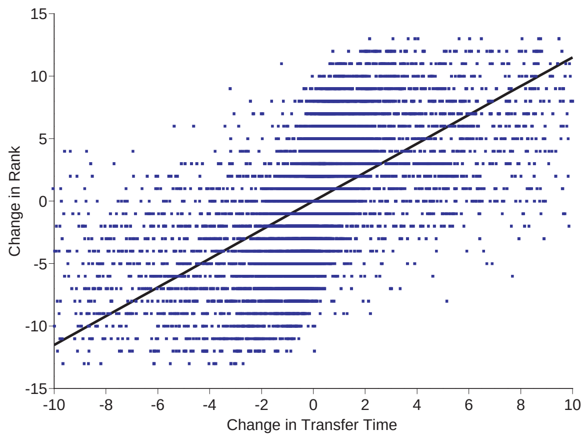


Figure 11: Changes in rank versus changes in transfer time (limited to +/- 10 seconds) for News/0 dataset. Note the inferiority of linear fit ($R^2 = 0.36$.) There is little relationship between changes in transfer time and changes in ranking.

it means that a client must consider all mirrors when choosing a server. From the data we have, we can build a working set for each client-document combination using a straightforward greedy algorithm: In each group of fetches, all servers that deliver good performance are marked. The number of marks that each server acrues over all groups is computed, and the server, s, with the highest total, is added to the working set. The groups where s exhibited good performance are discarded, and this procedure is repeated on the remaining groups. The algorithm terminates when there are no groups left.

Figure 12 shows the composition of the working sets for 10 data sets composed of the five documents from U. Mass's Apache data and the five documents from Washington U.'s Mars set. Each stripe from each column represents the proportion of time that one server offers good performance. For example, the first column of the graph shows the working set for the Wash. U. client's fetches of document 0 from the Mars sites. Each stripe in that column corresponds to one server. For purposes of this discussion, it does not matter which server maps to which stripe. What is significant is the size of each stripe, which shows how often the corresponding server is able to deliver good performance. The first stripe in the column shows that one server is good in almost 70% of its fetches. The next stripe represents a server that is good in a little more than 10% of the remaing fetches.

The distribution and number of stripes show that client sites do not have to consider every server in the mirror set to achieve good performance. Rather, a small number of servers can provide good performance for a significant fraction of all fetches. Looking at the Washington U. data, we see that for documents 1 through 4, the client can receive good performace over 90% of the time by considering only 2 servers out of the group of 20. For document 0, the client would need to consider 5 servers to achieve good performance more than 90% of the time. On the other hand, the client at U. Mass. requires more servers to achieve good performance when fetching from Apache servers. Seven servers are required for the first document while 5 are required for the other documents. This is a much higher proportion of servers than for the Washington U. client (7 out of 11 vs. 5 out of 20).

Figure 13 summarizes our findings over all documents. On average, less than half of all servers

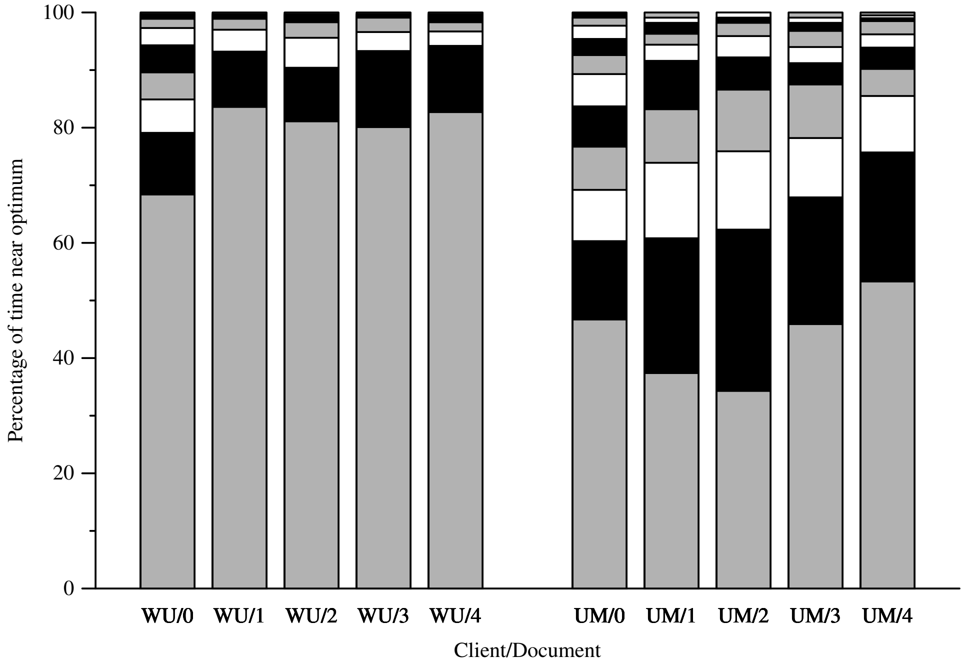


Figure 12: Working sets for two client-data combinations: Wash. U.'s Mars set and U. Mass.'s Apache

need to be considered to achieve good performance most (or even all) of the time. This result implies that when designing an anycast protocol, it is unnecessary to assume that all clients will need to contact all servers.

8 Server choice and document choice

The reader may have noticed that in Figure 12, the composition of working sets obviously varied from document to document. This seems to suggest that in some cases, a server that provides good performance for one document does not provide good performance for another document. Upon further examination, we have have found that this condition occurs frequently in both Apache and Mars data sets (News data is not applicable since it only includes one document). There is some property of a document upon which server quality is dependent. Unfortunately, we do not have enough data to determine which properties are significant, but we suspect that document size is a likely candidate.

Figure 14 shows how frequently the set of good servers for one document does not intersect with the set of good servers for another document in the Apache data set. The number at box i, j is calculated as the percentage of groups in which there was no possibility of achieving good performance fetching document j with any of the servers that would have given good performance in fetching document i. As in Section 7, we consider a "good" server to be capable of delivering a document within 10% of the fastest transfer time.

From Figure 14, we see that for any pair of documents, there were a significant number of

Doc.	Avg. for 90% Good	Avg. for 100% Good
Mars (20 Servers)		
0	3.44	8.57
1	2.67	5.83
2	2.56	5.83
3	2.67	5.67
4	2.22	5.60
Apache (11 servers)		
0	3.89	6.25
1	3.00	5.20
2	3.11	5.25
3	3.00	5.80
4	3.00	6.00
News (16 servers)		
0	2.44	5.88

Figure 13: Average (taken over all clients) number of servers required to achieve good performance in 90% and 100% of fetches

Document	0	1	2	3
1	35.37%			
2	45.33%	24.88%		
3	51.36%	25.61%	15.54%	
4	49.79%	26.34%	16.93%	11.32%

Figure 14: Proportion of groups in which two documents do not share any good servers. (Apache data, all clients aggregated.)

groups in which no single server provided good performance. For instance, in less than half of all fetches, a server provided good performance for both documents 0 and 3. The same trends are present in the Mars set, though the actual numbers are slightly lower.

We believe the dependence is more a function of document size than the specific documents being fetched, but further study using a larger variety of documents is required to verify this. We can explain the effect of document size on server choice if we assume that the network (and not the server) is the bottleneck. For smaller documents, the transfer time depends more on the round trip time between the client and server. The smallest documents fit in one or two packets so the client-server conversation lasts only a few round trip times. For larger documents, the amount of bandwidth available on the path between the client and server becomes the important factor as the network "pipe" is packed with as much data as possible. In this scenario, one property of a server (the round trip time between it and the client) would dominate for small documents and another propoerty (the throughput between the client and server) would dominate for large documents.

9 Implications for Anycast Protocols

The observations about the properties of mirror servers that we have discussed may be useful when designing anycast protocols. However, our measurements were made in the absence of an anycast protocol. The introduction of a systematic way to evaluate servers may alter the performance of the servers significantly. For example, the introduction of load balancing among servers may increase the correlation of performance among the servers. Still, our observations do supply a picture of the network that can be used as a starting point in designing a protocol.

In general, our results support an architecture similar to SPAND, proposed by Seshan et al. [23], in which previous performance results are cached on a campus-wide basis and are used to predict future performance. SPAND collects performance results continuously. However, because server rank changes do not depend on time significantly and since performance is predicted nearly as well by results collected days ago as by results collected hours ago, it may be possible for results on server rankings to be recorded infrequently.

Disappointingly, our conclusions from Section 6 indicate that a client cannot rely on changes in fetch times to a server to indicate changes in its rank with any great certainty. Had there been a stronger link between changes in transfer time and changes in rank, it might have been possible for a client to decide when a new search for a good server needed to be started purely based on its own local performance observations. It seems that using fetch times is most useful when results from fetches to multiple servers are compared.

For the News and Mars data sets, we found that most rank changes are small, implying that a client may assume with a reasonable amount of confidence that a server which delivered good performance during the last fetch will have acceptable performance during the next fetch, even if the two fetches are far apart in time. This may be a challenge for anycast protocols, since it might be the case that the overhead associated with finding a new server is larger than the performance gained from using the new server. For some reason, server performance seems to be less stable in the Apache set, perhaps indicating that a more proactive anycast protocol is necessary at least some of the time.

In Section 7, we argued that near-optimal performance can be achieved by considering relatively few servers. This implies that a client may be able to build a list of servers to ignore. The result is that server selection could be more efficient since there would be fewer servers to evaluate at any given moment. The client would build its list of servers to ignore by starting with the assumption that all servers should be considered and gradually pruning those that are consistently poor performers.

Finally, protocols should make allowances for picking a server based on the document that is being fetched. While we have not completely explored this area to determine which features of a document are most important in predicting server performance, it is apparent that aggregating performance results over an entire site's contents is likely to be a poor design choice. Similarly, a client should not rely on a single server to provide good performance for every document on a site.

10 Conclusion

We have presented measurements of the performance of replicated web servers which have ramifications for anycast protocol designs. We have found that though transfer times are highly variable, server ranks are fairly stable over time. Further, clients can receive good performance if they use a carefully chosen subset of a group of mirrors. Finally, we have seen that server performance varies with a client's choice of document. Though we suspect that document size is the determining factor, more study of this is necessary to reveal which characteristics of a document are significant. Other future work includes collecting longer traces and trying other mirror sets.

11 Acknowledgements

We would like to thank the client sites (Berkeley, Georgia Tech., ISI, the University of Kentucky, the University of Massachusetts, the University of Texas, the University of Virginia, and Washington University in St. Louis), who gave us guest accounts and tolerated our experiments. We would also like to thank the server administrators for their support and cooperation.

References

[1] M. Arlitt and C. L. Williamson. Web server worload characterization: The search for invariants. In *Proceedings of ACM SIGMETRICS '96*, 1996.

[2] Hari Balakrishnan, S. Seshan, M. Stemm, and R. H. Katz. Analyzing stability in wide-area network performance. In *Proceedings of ACM SIGMETRIC Conference on Measurement and Modeling of Computer Systems*, June 1997.

[3] Samrat Bhattacharjee, Mostafa H. Ammar, and Ellen W. Zegura. Application-layer anycasting. Technical Report GIT-CC-96/25, Georgia Institute of Technology, 1996.

[4] Samrat Bhattacharjee, Mostafa H. Ammar, Ellen W. Zegura, Viren Shah, and Zongming Fei. Application-layer anycasting. In *Proceedings of INFOCOM '97*, 1997.

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

[6] J. C. Bolot, H. Crepin, and A. Vega Garcia. Analysis of audio packet loss in the internet. In *Proceedings of NOSSDAV '95*, 1995.

[7] George E. P. Box, Gwilym M. Jenkins, and Gregory Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, 3rd edition, 1994.

[8] R. Cáceres, P. B. Danzig, S. Jamin, and D. J. Mitzel. Characteristics of wide-area TCP/IP conversations. In *Proceedings of ACM SIGCOMM '91*, 1991.

[9] Robert L. Carter and Mark E. Crovella. Dynamic server selection using bandwidth probing in wide-area networks. Technical Report BU-CS-96-007, Boston University, March 1996.

[10] M. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. In *Proceedings of ACM SIGMETRICS '96*, 1996.

[11] O. Gudmundson, D. Sanghi, and K. Agrawala. Experimental assessment of end-to-end behavior on internet. In *Proceedings of Infocom '93*, 1993.

[12] S. A. Heimlich. Traffic characterization of the NSFNet national backbone. In *Proceedings of ACM SIGMETRICS '90*, 1990.

[13] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, Inc., 1991.

[14] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic. In *Proceedings of ACM SIGCOMM '93*, 1993.

[15] B. A. Mah. An empirical model of http network traffic. In *Proceedings of Infocom '97*, 1997.

[16] J. C. Mogul. Observing TCP dynamics in real networks. Technical Report 92/2, Digital Equipment Corporation Western Research Lab, 1992.

[17] J. C. Mogul. Network behavior of a busy web server and its clients. Technical Report 95/5, Digital Equipment Corporation Western Research Lab, 1995.

[18] C. Partridge, T. Mendez, and W. Milliken. Request for comments 1546: Host anycasting service, November 1993.

[19] V. Paxson. Empirically-derived analytic models of wide-area TCP connections. *IEEE/ACM Transactions on Networking*, August 1994.

[20] V. Paxson. End-to-end routing behavior in the Internet. In *Proceedings of ACM SIGCOMM '96*, 1996.

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

[22] M. Schwartz, D. Ewing, and R. Hall. A measurement of Internet file transfer traffic. Technical Report CU-CS-371-92, University of Colorado at Boulder, 1992.

[23] Srinivasan Seshan, Mark Stemm, and Randy H. Katz. SPAND: Shared passive network performance discovery. In *Proceedings of USITS '97*, 97.

[24] M. Yajnik, J. Kurose, and D. Towsley. Packet loss correlation in the MBone multicast network. In *Proceedings of GLOBECOM'96*, 1996.