Performance Characteristics of Mirror Servers on the Internet

Andy Myers Peter Dinda Hui Zhang
Carnegie Mellon University
Pittsburgh, PA
{acm,pdinda,hzhang}@cs.cmu.edu

Abstract— **As a growing number of web sites introduce mirrors to increase throughput, the challenge for clients becomes determining 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 47 production web servers which mirror three different web sites. We have several interesting findings that may aid in the design of protocols for choosing among mirror servers. Though server performance varies widely, we have observed that a server's performance relative to other servers is more stable and is independent of time scale. In addition, a change in an individual server's transfer time is not a strong indicator that its performance relative to other servers has changed. Finally, we have 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.**

I. Introduction

Mirror servers, which serve replicas of popular data items, have been employed for many years on the Internet as a way to increase reliability and performance in the presence of frequent accesses by many clients. While mirroring can provide much higher aggregate throughput to a given data item, individual clients must choose a mirror server carefully to achieve reasonable performance. Unfortunately, only ad hoc mechanisms for choosing the appropriate mirror server are currently employed. However, a number of server selection mechanisms have been proposed. Partridge et al [1] have introduced a scheme called *anycast* that allows a client to automatically reach the replica of a server which is the smallest number of network hops away. Others [2], [3] 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.

In order to design an effective algorithm for mirror server selection, an understanding of the actual behavior of Internet mirror servers is necessary. To contribute towards this understanding, we have undertaken a large measurement scale study involving 9 clients and 47 mirror servers scattered throughout the United States. Although other studies of mirror server behavior have appeared in the literature before, we believe this is the first study of this scale. This paper presents a number of interesting properties that we have observed in the data collected. Our focus is on characterizing the performance an individual client receives when transferring documents from mirror servers. We wish to answer four questions:

- Does performance observed by a client vary across mirror servers?
- How dynamic is the set of servers that offer good performance?
- How is the probability that a server's performance will drop relative to other servers affected by time scale?
- Does a drop in a server's performance indicate it has become less likely to offer better performance than other servers?

To answer the first question, we have looked at the time required to retrieve a document from each mirror server of a site. We have found that the difference in performance between the best and worst servers is typically larger than an order of magnitude, and can grow larger than two orders of magnitude on occasion. This result shows that performance does indeed vary largely from one server to another.

The second question is an attempt to explore how dynamic server performance changes are. By counting the number of servers that a client must visit over time in order to achieve good performance, we can see whether the set of servers that offer good performance at any given time is small or large. We found that the set is usually fairly small, indicating less dynamic behavior.

The third and fourth questions concern mechanisms that could potentially be incorporated into a server selection system. In the third question, we consider whether there is some relationship between a server's performance relative to other servers and time scale. The fourth question considers the case in which a client has found a server that offers good performance relative to the other servers but then notices a drop in that server's performance. The question is whether or not that drop in performance indicates that the server's performance is no longer good. We found that large performance drops do indicate an increased likelihood that a server no longer offers good performance.

Finally, we will consider the effect of document choice on server choice. Though we assume that all mirrors of a server have the same set of documents, it might be the case that some factor such as document size or popularity would affect the performance of a server. We found that server choice is independent of document choice almost all the time.

To summarize, we have five main results:

- Performance can vary widely from one server to another.
- Clients can achieve near-optimal performance by considering only a few servers out of the whole group of mirrors.
- 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.
- Server choice is independent of document choice in most instances.

We discuss the implications of these results in Section IX.

A. Related work

Previous work on server selection techniques can be divided into four categories: network-layer server selection systems, application-layer selection systems, metric evaluation, and measurement studies. The first includes work dealing with finding

This research was sponsored by DARPA under contract numbers N66001-96-C-8528 and E30602-97-2-0287, and by NSF under grant numbers Career Award NCR-9624979 and ANI-9814929. Additional support was provided by Intel Corp. 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, or the U.S. government.

Client Site	Avg. time	Fetches	Failure rate
CMU	32.82 min.	54695	10.18%
Ga. Tech.	23.78	60021	11.55%
ISI	36.52	53200	22.13%
U. C. Berkeley	32.55	55062	4.62%
U. Kentucky	31.23	55091	12.76%
U. Mass.	70.56	36542	10.95%
U. T. Austin	39.56	51640	4.70%
U. Virginia	19.32	62405	28.88%
Wash. U., St. Louis	23.27	62187	1.96%

Fig. 1. Average time for one group of fetches, number of fetches completed, and failure rate for each client site.

the closest server in terms of number of network hops or in terms of network latency [1], [4], [5], [6], [7], [8], [9]. The second consists of systems that take application performance metrics into account [3], [10], [11], [12], [13], [14], [15]. Most of these systems use a combination of server load and available network throughput to select a server for a client. The third category consists of evaluations of server selection metrics [2], [16], [17], [18]. These studies propose new metrics and test them experimentally.

The fourth category, which includes this work, consists of studies that characterize the behavior of existing mirror servers in order to draw conclusions about the design of server selection systems. Bhattarcharjee et al [19] measured "server response time," 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. Carter and Crovella [2] measured ping times and hop counts to 5262 web servers to determine 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.

There have been several other web-related measurement studies. Balakrishnan et al [20] analyzed a trace of web accesses to determine how stable network performance is through time and from host to host. Gribble and Brewer [21] looked at users' web browsing behavior, exploring server response time, burstiness of offered load, and the link between time of day and user activity. Cunha et al [22] also collected user traces via a customized version of Mosaic and looked at a number of factors including document size and popularity. Arlitt and Williamson [23] searched for trends present in a variety of different WWW workloads based on server access logs. Finally, Crovella and Bestavros [24] have found evidence for self-similarity in WWW traffic.

The rest of this paper consists of a description of our data collection system (Section II), a general picture of the data we collected (Sections III and IV), a discussion of our findings (Sections V through VIII), implications of our results (Section IX), and conclusions (Section X).

II. Data collection methodology

At each of nine client sites where we had guest accounts (listed in Figure 1) a perl script periodically fetched documents from each server in three sets of mirrored web sites (the Apache Web Server site, NASA's Mars site, and News Headlines) listed in Figure 2. 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

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

Fig. 2. Servers visited

	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

Fig. 3. URLs of documents fetched Mars and Apache servers.

(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). We chose these sites in order to capture three different ranges of site content update frequency: the Apache site's content changed on the order of weeks; the Mars site, on the order of days; and the News site, on the order of minutes.

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 site. 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 refer to the process of visiting all servers and collecting all documents once as a *group* of fetches.

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 load our fetches created on client and server sites. A typical group of fetches involved transferring more than 60 MB 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 roughly half the original average bandwidth.

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 killed. We chose five minutes as a compromise between achieving a complete picture of a server's behavior and forcing groups of fetches to finish in a reasonable amount of time. The observable effects of such a short timeout were a slightly higher failure rate, especially among larger documents. Possible causes for timeouts are network partitions, client errors (lynx might have frozen), server errors (the server might have stopped providing data), or 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 so 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 other servers, fetches lasting five minutes were considered failures.

While our fetch scripts were running, there were multiple occasions 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.

A. 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 because of variations in server load and network congestion, the distribution of fetch interarrivals to a single server from a client was extremely 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 [14], [20] 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.

DNS caching effects could also potentially bias our results. Depending on the DNS workload at a given client site, DNS entries for the servers in our study may or may not remain in the local cache from one group of fetches to another. In fact, cache entries could even be purged within a group of fetches. The DNS lookups added a potentially highly variable amount of time to each fetch we performed. Performing the lookups separately would have been possible, but less realistic.

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.

III. 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. The script acted this way because we assumed that any machine crash and reboot would take

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

a significant amount of time. Therefore, a new group was started to avoid a group's fetches from stretching over too long a period of time. 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 were classified as failures (because timeouts and improper amounts of data returned). 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 high failure rate.

A. Ranks

Throughout this paper, we use *rank* to compare servers' performance. In this section we explain how ranks 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 II-A, 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 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 increase in transfer time vs. server rank for document 4 of the Mars data set. It was produced by averaging the differences of all pairs of servers with ranks i and $i-1$ in each group. The graph shows a definite rise in transfer time as rank increases. For example, we 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.

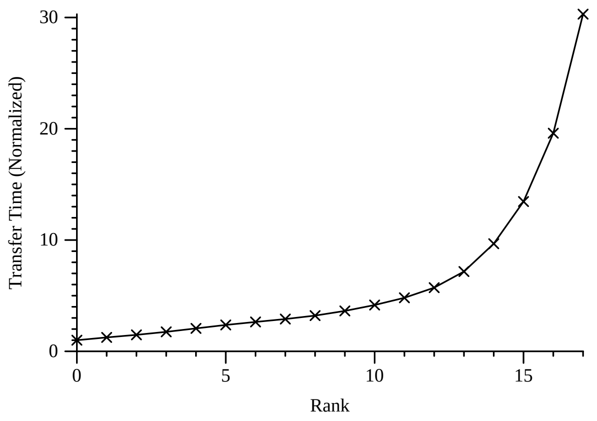

Fig. 4. Average amount of separation between rank values for Mars servers, document 4, all clients aggregated.

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.

Figure 4 also answers our first question about whether performance varies across mirror servers. A factor of 30 separates the transfer times of the best and worst servers. Even the top 10 servers' performance varies by a factor of 5. We noticed the same or more exaggerated ranges of performance for other documents and sites. Performance definitely varies widely across mirrors of a site.

IV. 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 variability. Second, transfer times are well fit by a Weibull 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.

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 observed 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

Dataset/Doc	Transfer time (seconds)				
	Mean	StdDev	Median	Min	Max
Apache/0	1.9632	5.8366	.7	0.1000	230.5100
Apache/1	3.9112	7.9753	2	0.3800	297.700
Apache/2	3.2929	6.3993	1.7	0.3000	293.9000
Apache/3	15.4776	18.2385	10.7	1.3000	299.9000
Apache/4	23.1960	22.9257	17.9	2.2000	298.2000
Mars/0	1.5416	4.6808	0.7	0.1000	296.6000
Mars/1	2.6929	6.5319	1.3	0.1000	292.6000
Mars/2	5.8062	9.4102	3.3	0.3000	290.5000
Mars/3	8.7380	12.3967	5.3	0.6000	297.3000
Mars/4	19.9019	23.5427	13.9	1.6000	298.2000
News/0	3.8185	11.8028	1.06	0.1200	638.9800

Fig. 5. Summary statistics of transfer time.

document. In some cases, particularly slow fetches can result in normalized transfer times exceeding 300 seconds. This is rare.

An interesting observation can be made about ranks. Although ranks are highly variable, this does not bode disaster for server selection algorithms. A random selection is likely to result in an average ranked server, which by Figure 4 would seem to indicate reasonable performance. 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.

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 [25, 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. Generally, log-normal was much better than the earlier distributions. This result agrees with Balakrishnan et al [20], who also found that a single client's observed throughput can be modeled reasonably well by a log-normal distribution.

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. The transformed data are well fit with an exponential distribution, thus the original data is distributed according to a Weibull distribution.

It important to note that because transfer times were artificially truncated at 5 minutes, we do not have an accurate picture of the full 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 Weibull distribution may not fit this data as well as it seems to.

V. The time scale 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 time scale 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

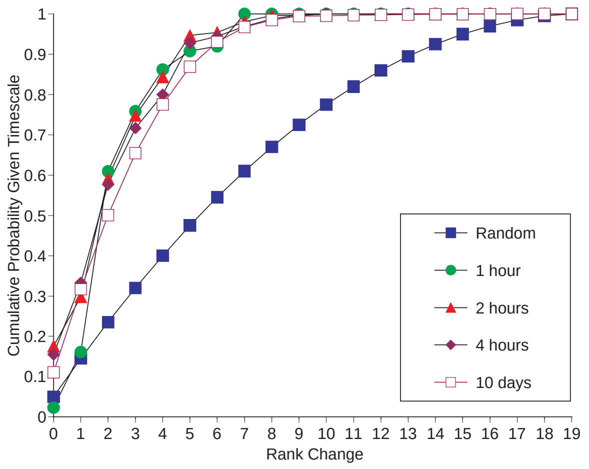

Fig. 6. $P[|r_{t+w} - r_t| \leq R \mid sample\ period \leq w \leq W]$ for Mars/1 dataset, plotted for several different values of W. R is the rank change and W is the maximum time period. The other Mars plots and the News/0 plot are similar.

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 time scales would be via frequency domain or time series analysis [26]. However, as we discussed in Section II, 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 complex and since signal reconstruction from non-periodic 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 occurrences 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 6 shows a representative plot of the cumulative probability for the Mars/1 dataset. The way to read the plot 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, we see that for time scales of two (or fewer) hours, rank changes of four

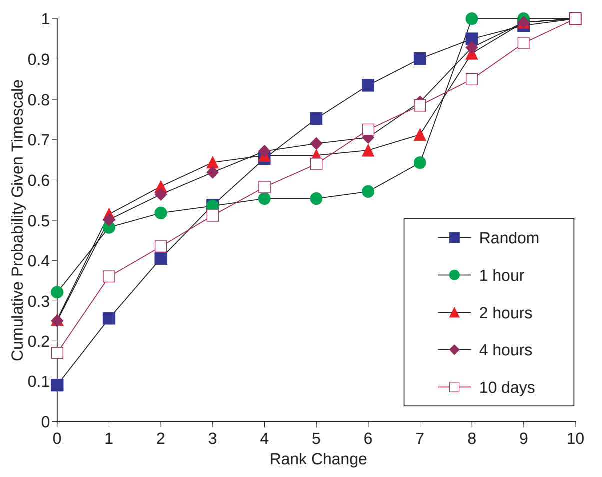

Fig. 7. $P[|r_{t+w} - r_t| \leq R \mid sampleperiod \leq w \leq W]$ for Apache/1, plotted for several different values of W. Other Apache plots are roughly similar, and all Apache plots differ significantly from Mars or News plots.

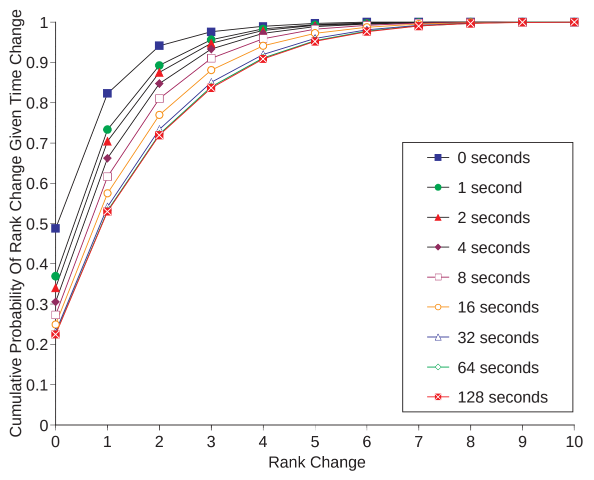

Fig. 9. Cumulative probability of rank change given changes in transfer time less than D ($P[|r_{t_i} - r_{t_j}| \leq R \mid |d_{t_i} - d_{t_j}| \leq D]$) for Apache/4, plotted for several different values of D. All other plots are similar.

(or fewer) occur with probability 0.85. The plot also includes the curve that would result if rankings were simply uniformly distributed random permutations.

It is clear from the graph 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 graph also shows 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 do indeed 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 graph of Figure 6 is representative for the Mars and News datasets. Unfortunately, the Apache data shows very different behavior, as can be seen in Figure 7, 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 distinguishing characteristic of the Apache sites is that they tend to run non-commercial 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 discrepancy, however.

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

The approach we present here is to estimate the cumulative probability of rank changes over increasing changes in observed transfer times. We have also examined the cumulative probability of rank changes over increasing *percentage* changes in transfer time, and we have found results similar to those presented in this section.

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 8 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 symmetric about zero. For this reason, we concentrate on absolute changes in transfer time and rank.

After taking absolute values, we count occurrences 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 V that changes in rank are virtually independent of time scale.

Figure 9 shows a representative plot of the cumulative probability for the Apache/4 dataset. The plots for all of the datasets are similar. The way to read the plot 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, we see that for a transfer time change of 128 seconds or less, about 90% of rank changes are of four 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 the right (toward larger rank changes.) However, the difference is slight. For example, a rank change of three or smaller is 90%

	Changes in transfer time (seconds)					Changes in rank				
Dataset/Doc	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

Fig. 8. Summary statistics of changes in transfer time and changes in corresponding ranks.

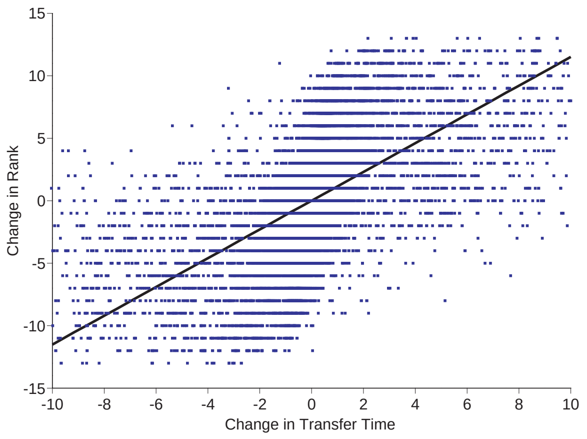

Fig. 10. Changes in rank versus changes in transfer time (limited to +/- 10 seconds) for News/0 dataset. Note the inferiority of linear fit ($R^2 = 0.36$.) There is little relationship between changes in transfer time and changes in ranking.

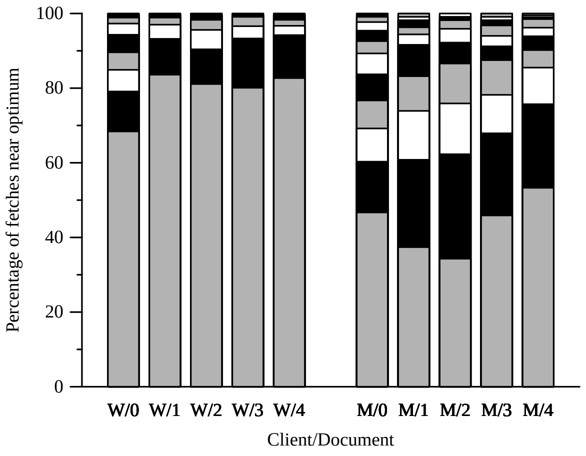

Fig. 11. Server sets for two client-data combinations: Wash. U.'s Mars set and U. Mass.'s Apache

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 10 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. Examining each client and server pair of all the datasets individually, we find that only 10% of combinations yielded R^2 values greater that 0.5, fewer than 1% yielded R^2 values greater than 0.8, and the highest R^2 value was only 0.88. Clearly, there is only a limited relationship between changes in transfer time and changes in rank.

VII. Small server sets

The observation in Section V 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 only 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 best transfer time for the same document observed in the same group of fetches.

We define a *server set* for client C and document D to be the minimum subset of a site's mirrors such that for any group, at least one server in the server set can deliver D to C with good performance. If a server set contains all the mirrors, it means that a client must consider all mirrors when choosing a server. From the data we have, we can build a server 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 accrues over all groups is computed, and the server, s, with the highest total, is added to the server set. The groups where s exhibited good performance are discarded, and the procedure is repeated on the remaining groups. The algorithm terminates when there are no groups left.

Figure 11 shows the composition of the server 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 server 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 remaining

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

Fig. 12. Average (taken over all clients) number of servers required to achieve good performance in 90% and 100% of fetches

	Document j				
SS i	0	1	2	3	4
0	0.55%	9.32%	9.86%	10.68%	8.77%
1	4.11%	0.00%	0.00%	0.55%	0.27%
2	4.11%	0.00%	0.00%	0.55%	0.27%
3	6.85%	0.82%	0.27%	0.00%	0.00%
4	6.85%	0.82%	0.27%	0.00%	0.00%

Fig. 13. Percentage of time that good performance is not achieved using the top 5 servers from the server set of document i (SS i) to fetch document j.

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 performance 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 12 summarizes our findings over all documents. On average, less than half of all servers need to be considered to achieve good performance most (or even all) of the time. This result implies that when designing a server selection mechanism, it is unnecessary to assume that all clients will need to contact or evaluate all servers.

VIII. Server choice and document choice

The reader may have noticed that in Figure 11, the composition of server sets obviously varies 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. However, further examination reveals that document choice has at best a weak effect on server choice.

Recall that a server set is the *smallest* set of servers that provide good performance for a given client. Other servers not in the server set could provide good performance at any given moment. For example, there are cases in which more than one collection of servers can be a server set. If two servers, A and B, provide good performance at exactly the same moment, then two server sets are possible: one using A and the other using B. Thus, it is unwise to rely on apparent differences in server sets as an indicator of differences in server performance.

Figure 13 shows how using one document's server set to fetch another document affects performance. The table was built by counting how often the top 5 servers from document i's server set (SS i) are able to offer good performance for document j for every $i, j \epsilon [0, 4]$. Although this data is generated from the Mars data at client site U. Va, all other combinations of clients and web sites produced similar results. The entry at (i, j) in the table is the percentage of fetches for which the server set for document i was **not** able to provide good performance for document j. For example, we can see that using the server set for document 4 to fetch document 1 would lead to good performance in over 99% of fetches.

We used only the top 5 servers from each server set so that all sets of servers considered would be the same size. Server sets for documents 2 through 4 only contained 5 servers, so they were unaffected. Document 0's server set, however, contained 7 servers. The most immediate effect is that in the table above, the (truncated) server set for document 0 failed to provide good performance 0.55% of the time.

Measuring how well one document's server set would do to fetch another is a much more reasonable way to judge the differences in server performance among documents. It can directly show how often a server identified as good for one document is not actually good for another document. In Figure 13, we can see that most often, performance remains good across server sets. Ignoring data from the first row and first column, we see that instances when one document's server set does not offer good performance for another document are very rare.

Looking at the table's first row and the first column, which correspond to server set 0 and document 0 respectively, we see that good performance is achieved less frequently. The servers which offer good performance for document 0 are at least partially different from the servers that offer good performance for other documents. This indicates that there might be some link between document choice and server choice. In all client-site combinations, we observed that the first document had a noticeably different set of good servers than the other documents.

In both the Apache and Mars data, the first document is also the smallest (about 2 KB). 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 property (the throughput between the client and server) would dominate for larger documents.

Regardless of the cause, the effect is not extremely significant. First of all, at most 11% of fetches were adversely affected by the difference in server sets. In these fetches, the increase in transfer time was less than 25% above optimal on average. Also note that these performance penalties are on top of a rather small transfer time (about 1 second), so the actual penalties are on the order of hundreds of milliseconds. Thus there is little cause for concern over using only one server set for all document sizes will lead to bad performance.

IX. Implications for server selection systems

The observations about the properties of mirror servers that we have presented may be useful when designing server selection systems. However, our measurements were made in the absence of a selection mechanism. The introduction of a systematic way to evaluate servers may alter the performance of the servers significantly. For example, if all clients employ a load balancing algorithm, the correlation of performance among the servers may increase. Still, our observations do supply a picture of the network that can be used as a starting point in designing a system.

We have pointed out that the difference in performance from one mirror server to another is quite significant. This implies that choosing the right server has the potential to significantly improve client performance. We have also seen that most server sets usually contain more than one server, indicating that the best server for a given client changes over time. Server selection needs to take place periodically to achieve good performance. But because server sets are also usually small, the server selection task is potentially a very lightweight operation.

We have observed that server rank changes do not depend significantly on time scale, implying that a ranking of servers from two hours ago is as useful as a ranking from two days ago. In other words, all performance results older than an hour are approximately equally useful. Because of our experimental design, we cannot say anything about performance results younger than an hour.

For the News and Mars data sets, we have 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. For these data sets, the benefits of server selection may be outweighed by the cost of evaluating servers. However, this does not hold for the Apache set, where ranks are less stable.

We have found a weak link between a change in a server's performance and a change in the server's rank. If the performance that a server can offer a client degrades massively, then it can be inferred that the server's rank has dropped and a new server should be selected for the client. However, for smaller performance drops, we cannot assume that a corresponding drop in rank has taken place.

Finally, protocols probably do not have to make allowances for picking a server based on the document that is being fetched. While we have noticed that there is a difference between the good servers for the smallest Mars and Apache documents and other documents' good servers, the difference in performance, both in relative and absolute terms, is not very large.

X. Conclusion

We have presented measurements of the performance of replicated web servers which have ramifications for server selection system designs. We have found that performance observed by a given client can vary widely from mirror to mirror. However, the set of servers that a client must visit to achieve good performance is fairly small. Once a client has found a good server, neither time scale over which the server has been good nor moderate changes in server performance are good indicators about when to begin searching for a new good server. However, a large performance drop does signal a client that a server's rank is likely to have changed. To further substantiate and expand our conclusions, future work includes collecting longer traces, trying other mirror sets, and exploring shorter time scales.

The data collected for this study is available on the World Wide Web at http://www.cs.cmu.edu/~acm/research/anycast.html.

XI. Acknowledgments

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] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service," IETF RFC 1546, November 1993.

[2] R. L. Carter and M. E. Crovella, "Dynamic server selection using bandwidth probing in wide-area networks," Tech. Rep. BU-CS-96-007, Boston University, March 1996.

[3] S. Bhattacharjee, M. H. Ammar, E. W. Zegura, V. Shah, and Z. Fei, "Application-layer anycasting," in *Proceedings of INFOCOM '97*, 1997.

[4] P. Francis, S. Jamin, V. Paxson, and L. Zhang, "Internet distance maps (IDMaps)," Available at http://idmaps.eecs.umich.edu/.

[5] Cisco, "DistributedDirector," Available at http://www.cisco.com/warp/public/751/distdir/dd_wp.htm.

[6] E. Basturk, R. Engel, R. Haas, D. Kandlur, V. Peris, and D. Saha, "Using network layer anycast for load distribution in the Internet," Tech. Rep., IBM T.J. Watson Research Center, 1997.

[7] P. Francis, "A Call for an Internet-wide Host Proximity Service (HOPS)," http://www.ingrid.org/hops/wp.html.

[8] D. B. Johnson and S. E. Deering, "Reserved IPv6 subnet anycast addresses," IETF Internet Draft `draft-ietf-ipngwg-resv-anycast-01.txt`, 1998.

[9] B. N. Levine and J. J. Garcia-Luna-Aceves, "Improving Internet multicast with routing labels," in *Proceedings of ICNP '97*, 1997.

[10] BrightTiger, "ClusterCats," Available at http://www.brighttiger.com/products/default.htm.

[11] WindDance Networks, "WebChallenger," Available at http://www.winddancenet.com/products/challenger.html.

[12] J. Rosenberg and H. Schulzrinne, "Internet telephony gateway location," in *Proceedings of INFOCOM '98*, March 1998.

[13] J. Rosenberg, H. Schulzrinne, and B. Suter, "Wide area network service location," IETF Internet Draft `draft-ietf-svrloc-wasrv-01.txt`, 1997.

[14] S. Seshan, M. Stemm, and R. H. Katz, "SPAND: Shared passive network performance discovery," in *Proceedings of the 1997 USENIX Symposium on Internet Technologies and System (USITS)*, 97.

[15] C.Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, and D. Culler, "Using smart clients to build scalable services," in *Proceedings of USENIX '97*, January 1997.

[16] Z. Fei, S. Bhattacharjee, E. W. Zegura, and M. H. Ammar, "A novel server selection technique for improving the response time of a replicated service," in *Proceedings of INFOCOM '98*, March 1998.

[17] J. D. Guyton and M. F. Schwartz, "Locating nearby copies of replicated Internet servers," Tech. Rep. CU-CS-762-95, University of Colorado at Boulder, 1993.

[18] M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek, "Selection algorithms for replicated web servers," in *Proceedings of the Workshop on Internet Server Performance '98*, June 1998.

[19] S. Bhattacharjee, M. H. Ammar, and E. W. Zegura, "Application-layer anycasting," Tech. Rep. GIT-CC-96/25, Georgia Institute of Technology, 1996.

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

[21] S. D. Gribble and E. A. Brewer, "System design issues for Internet middleware services: Deductions from a large client trace," in *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, 1997.

[22] C. R. Cunha, A. Bestavros, and M. E. Crovella, "Characteristics of WWW client-based traces," Tech. Rep. BU-CS-95-010, Boston University, 1995.

[23] M. Arlitt and C. L. Williamson, "Web server workload characterization: The search for invariants," in *Proceedings of ACM SIGMETRICS '96*, 1996.

[24] M. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and possible causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835–846, December 1997.

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

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