Evaluation of a Novel Two-Step Server Selection Metric*

Katrina M. Hanna† Nandini Natarajan Brian Neil Levine
Dept. of Computer Science, University of Massachusetts, Amherst, MA 01003
{hanna, nnataraj, brian}@cs.umass.edu

Abstract

Choosing the best-performing server for a particular client from a group of replicated proxies is a difficult task. We offer a novel, two-step technique for server selection that chooses a small subset of five servers, and isolates testing to that subset for ten days. We present an empirical evaluation of both our method and previously proposed metrics based on traces to 193 commercial proxies. We show that our technique performs better than any of the other metrics we studied — often one to two seconds better for a one-megabyte file — while requiring considerably less work over time. Metrics such as round-trip time and tests using small files usually select servers that are two to three times worse than the best server. Network-layer metrics such as minimizing router and autonomous system count poorly predict which server provides the best performance. These metrics often select servers with transfer times four to six times that of the best-performing server.

1 Introduction

Sites providing web content, multimedia streaming, networked gaming, or other Internet services commonly need to scale to large client bases. One solution often adopted is to replicate the server in numerous locations in the Internet. Presenting numerous mirror servers to a client results in the difficult problem of finding the server that will perform best. Techniques for *server selection* allow a client to transparently choose one of a set of known replicated servers — hopefully the best server. Proposals for implementing server selection and the related problems of anycast services [20] and Internet distance map creation [12, 8] are typically based on specific *metrics*, such as hop count or round-trip time.

In current practice, selection of a server from a group of proxies commonly requires manual choice based on geographical labels, though this has no correlation with network distances [4]. Commercial mirror selection services are commonly based on DNS modifications [1, 11]. Such commercial solutions use proprietary metrics and techniques, require costly Internet-wide infrastructure deployment, and have been shown to not select the best server in a consistent manner [13].

We address the following fundamental question: *what metric for server selection results in the best performance for clients?* We propose a novel technique called *ping-random*, which is a two-step process for server selection. The first step isolates a subset of five well-performing servers and retains the subset for a period of 10 days; during the 10-day period, server selection is random, which requires significantly less testing than previous approaches. We compare our technique with four previous approaches to server selection through experimentation and analysis: minimizing router counts, autonomous system (AS) counts, round-trip times, and transfer times of small test files. Our results indicate that:

- Our technique of *ping-random* can efficiently select servers with very good performance; metrics based on router hops, AS hops, round-trip times, and small files perform worse.
- Our technique works over a variety of server population sizes from 20 to 200.
- Recent proposals for the use of network-layer metrics perform very poorly as predictors of the best server from which to retrieve files;
- The best servers maintained very good performance over time, the average server did not. This corroborates our ping-random result.

We also examined the effects of large numbers of clients using our selection technique concurrently. Our method provides some load-balancing and does not result in client oscillation among servers.

This paper is organized as follows. Section 2 reviews previous work on anycast protocols and studies of metrics for server selection. Section 3 presents our experiment and methodology. Section 4 discusses our comparison of selection metrics. Section 5 provides characterizations of top servers. Section 6 offers our concluding remarks.

*This work is supported in part by a grants from Sprint Advanced Technology Labs and a grant from the National Science Foundation under award EIA-0080119.

†Katrina Hanna's work is supported in part by a National Science Foundation Graduate Research Fellowship.

1092-1658/01 $17.00 © 2001 IEEE

2 Background

Server selection techniques can meet a number of objectives. Our goal is to design and evaluate a metric that allows a client to choose the best-performing server from a set that provide a needed application-layer service; in this case, transferring a requested document to the client. We do not focus on techniques for processor load-balancing or resource discovery in this paper. We focus instead on situations where network conditions are the constraints, where all mirror sites are known, and are exact replicas.

2.1 Previous Work

Past work in server selection can be roughly divided into protocols based on specific metrics, studies of how well metrics can perform, and characterizations of server performance. Other work attempts to obviate the need for selection by proposing that files be retrieved in parallel from all available servers.

Protocol work for server selection, often in the context of anycast [20], has generally been proposed as a new network layer routing or application layer service. Proposed network-level protocols [6, 15] — most recently the Global Internet Anycast protocol (GIA) [14] — focus on discovering the best server by minimizing network distances of some type. As we show, these approaches are not good predictors of file transfer times and can result in problems balancing the client load across servers. Application-layer protocols [7, 2] consider combinations of network and server performance in selecting the best server.

In this paper, we examine a new selection metric and compare it to others. We expect our selection metric would be applied to application-level architectures; e.g., [2]. Moreover, we believe our contributions can increase the accuracy of Internet Distance Maps [12, 8].

Several recent proposals improve file transfer performance by retrieving content in parallel from several servers [3, 21], which reduces the need for server selection. We believe such protocols result in clients receiving an unfair share of bandwidth during TCP transfers. As congestion on a link grows, AIMD congestion control in TCP ensures that all distinct flows receive roughly equal bandwidth at a bottleneck. Clients downloading in parallel invalidate the assumptions of the AIMD algorithm.

2.1.1 Comparison to Past Experiments

In this paper, we compare our ping-random metric with previous metrics. We also characterize the performance of the servers in our study in order to gain insight into these comparisons. There have been several past studies on server selection metrics [4, 22, 19, 7, 2] and characterizing server performance [17]. Where those studies produced results that differ from ours we comment appropriately.

Table 1 summarizes those experiments in comparison to ours. We felt performing a new study was justified simply in order to compare our proposed new metric. However, the size of our experiment also differentiates our study from those done in the past.

While some studies share some of our experiment's features, no single previous study has all these qualities at once. Our experiment has the longest measurement period: 41 days. We used larger file sizes than most, up to 1 Meg, and compared many selection metrics at once. Our study uses actual transfer times for files of varying sizes where some have used estimates or not considered transfer times at all. We are also the first to study both metrics and server performance characteristics together.

We increase the number of servers by an order of magnitude over most studies to 193. Obraczka and Silvi recorded results from 601 servers, but did not study actual transfer times. Their focus was on the correlation between hop count and round-trip time. This may be important for streaming media applications, but it is likely that our calculation of the correlation between hop count and transfer times is more applicable to supporting file transfer applications. More importantly, we believe our measure of file transfer times resulting from these policies (Figs. 1–2) are more telling and reliable than correlation measures.

In a related study, Myers et al. [17] showed that there may be consistency in the rankings of servers even when exact performance changes between sessions. We found the rankings of the average server to be less stable than reported in their study. However, as we discuss in Section 5 top ranked servers were considerably more stable in ranking than both the average server in our study and in Myers et al. That result motivated our work on the two-step ping-random scheme that we propose and analyze in this paper.

3 Experimental Setup and Methodology

Our experimental setup included six client machines located at the Univ. of Massachusetts, the Univ. of North Carolina, the Univ. of Delaware, Purdue Univ., the U.C. Santa Cruz, and the Univ. of Southern California (Table 2). Each of these clients interacted with 193 servers in the tucows.com web mirror network. To avoid inter-continental links, we used servers located in the U.S. and Canada only.

Data was collected via a script that ran continuously at each client over the course of 41 days, from September 30 – November 9, 2000. (All logs are available at http://signl.cs.umass.edu/logs.) The script was implemented in Tcl/Expect. At each *run* of the script, the client collected several types of data regarding the characteristics of the network path to each server:

- a series of 5 ICMP pings;
- a traceroute;
- transfer times of files with approximate byte sizes of 10k, 30k, 100k, 250k, 500k, 750k, and 1M.

We then explored several ways to use such data for server selection. Ping results were used to evaluate the performance of selection schemes that minimize round-trip

Study	Client domains	Servers	Metrics compared	Measured	Days
Myers et al. [17]	9	20,16, and 11	None	File transfer	21 days
Sayal et al. [22]	1	5 and 50	Hop,ping	Http-request RTT	2000 requests
Obraczka and Silva [19]	4	601 (worldwide)	Hop,AS,ping	Round trip latency	6 days
Fei et al. [7]	2	4	Hop,small files,server push	File transfer	100 requests
Bhattacharjee et al. [2]	2	7	Small files, push from server	File transfer	Every 3 minutes for 13 hours
Carter and Crovella [4]	1	10	Hop,geographic, ping, trans. time estimate	File transfer	Several hours
This study	6	193 (US & CA)	Hop,AS,ping,10k files ping-selection	File transfer	Every 3–5 hours for 41 days

Table 1. Summary of some relevant past work.

times. Using traceroutes, we computed both router hop counts and autonomous system counts. We used these counts to characterize the performance of schemes that minimize either metric. AS-hops were calculated by querying a whois database.

The disparity in the numbers of runs completed by clients is a result of variation in the time required for each run. This is due to variations in the clients' system resources as well as the quality of their connections to the Internet. Rarely were runs aborted due to client failure or disconnection from the network (Table 2).

For much of the analysis in this paper, we considered the whole set of 193 servers; we also considered subsets. For example, we viewed our experiment as ten separate experiments of about 20 servers. The servers in each subset were created by simply partitioning the list of servers into groups as they appeared in order in the script that each client ran: the first 20 in the first set, the second 20 in the second set, and so on. We also consider four experiments of about 50 servers and two experiments of about 100 servers in order to measure the effects of different server populations. We refer to a group of n servers as an *n-subset*. Conducting the analyses on different subsets and averaging the results of the experiments increased our confidence that results were not due to one particularly good server for a particular client.

Network conditions may have changed from the beginning to the end of a run during the time it took to contact 193 servers from each client. However, accessing even small numbers of servers in parallel would have the transfers competing with each other for bandwidth. The examination of subsets of servers addresses this problem. Our results were consistent for 10 sets of 20 servers, which took on average, 9–15 minutes to complete. As we show in Section 4, our results for our selection method remain stable over periods of ten days without re-testing, so we believe short-term effects were not significant. Additionally, the consistency of results across subsets leads us to conjecture that time-of-day effects were not significant either.

We contacted the servers directly, avoiding any redirection from the main tucows site. It is possible that ISPs hosting a server employed DNS rotation techniques to map a textual address to more than one local server, but our focus on client performance would not be affected.

Client	File Downloads	Failure Rate
UNC	416,168	4.45%
Purdue	337,379	4.96%
U. Delaware	330,685	4.94%
U. Mass	164,843	6.22%
UC Santa Cruz	367,292	4.36%
USC	297,658	4.44%

Table 2. Client sites and data characteristics.

4 Server Selection Metrics

Recall that our goal is to compare our proposed metric with others to determine which is able to predict the server that will provide the best performance for the lowest cost. For the file transfer applications we wish to support, we define *best* performance as follows. As described in the previous section, each client periodically performs a *run* of measurements, contacting each of the 193 servers and performing five pings, a traceroute, and a series of downloads of given file sizes. For a each file size, the "best" performance is the minimum download time for that file size from among the 193 contacted for that measurement run. We compared metrics which select servers based on

- *ping-random*,
- round-trip times,
- transfer times of a 10k file,
- hop counts,
- AS counts,
- random selection,

Ping-random is a two-step server selection process that we designed: first, a subset of five well-performing servers is selected; second, a server is selected from among the servers in that subset. The subset is isolated based on

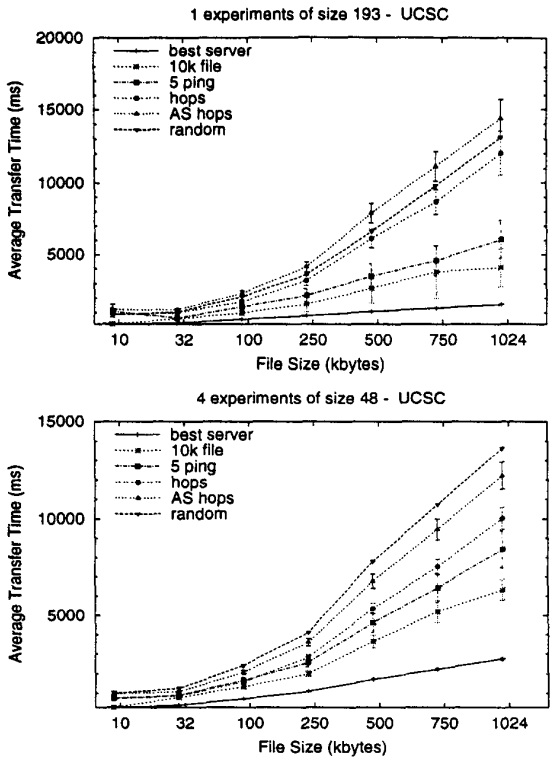

Figure 1. Performance of server selection metrics (top: UCSC, 193-set; bottom: UCSC, 48-subset).

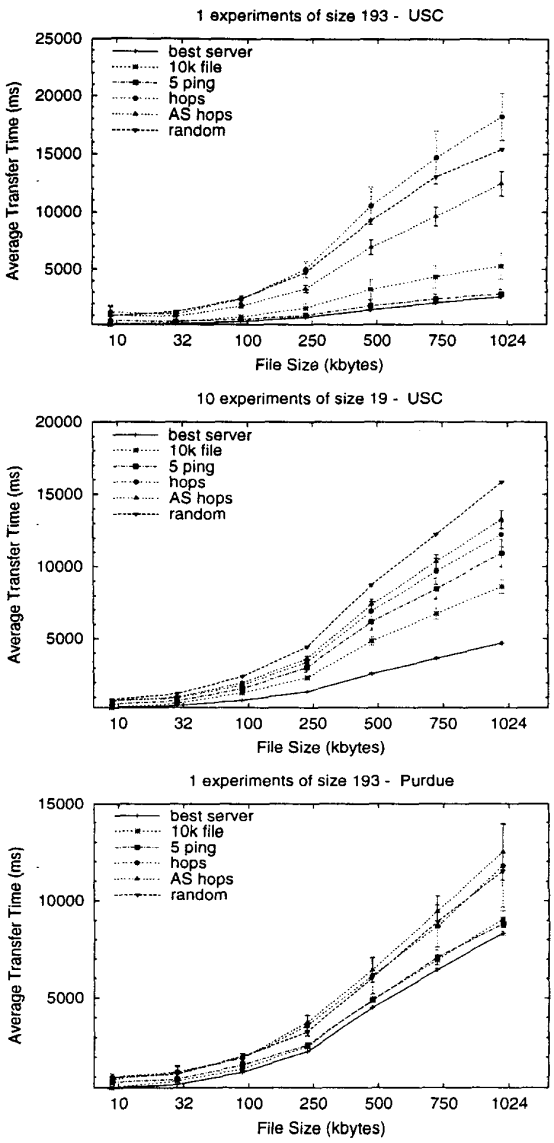

Figure 2. Performance of server selection metrics (top: USC, 193-set; middle: USC, 19-subset; bottom: Purdue, 193-set).

ping times and retained for a 10-day period. Over this 10-day period, a server is chosen at random for a particular file transfer. Methods other than random can be used for this secondary selection—we evaluate other possibilities in Section 4.1.3.

Previous work on the performance of mirrors [17] has shown the past performance of a server is often a reliable predictor of the future performance. In Section 5 we show that servers with the best transfer times are more stable than the average server. This lead us to design the ping-random technique and is an explanation of its success.

Our study indicates that ping-random performs best and requires the least work of all metrics. Our results are presented below. RTT times and transfer times of small files give some indication of server performance, but are not always accurate. Network-layer metrics—minimizing hop count and AS count—perform poorly as predictors of file transfer times, often as poorly as a random selection.

The performance differences among metrics was not distinguished for small files. Also, the time spent testing servers is less justified for small files. Thus, our results are most applicable for larger file transfers, and when clients download many files from the same site, e.g., with persistent and pipelined HTTP connections [16].

4.1 Dynamic Selection Methods

In this section, we detail our results for round-trip times (RTT), small file transfers, and our ping-random technique.

Determining the RTT to a remote server using an ICMP echo request is normally a simple and quick operation. RTTs, usually measured by pings, have the additional advantage over hop counts and AS counts of being responsive to network conditions. In fact, using RTTs appears more commonly than other metrics, with the exception of man-

ual, geographic selection. For example, Napster, a popular peer-to-peer system measures round-trip times to aid users in selecting the peer from which to retrieve a file.

Our results for some clients[1] are shown in Figs. 1–2[2] for a variety of metrics, each represented with a different line on the graph. The y-axis of the graphs represent the average transfer time of the servers selected by a metric; the performance of the server selection for different file sizes are shown along the x-axis. Error bars represent a 95% confidence interval of the averages.

In our experiment, clients sent five pings in immediate succession to servers. The RTT metric is better-correlated to actual file transfer time than hop count or AS count. Moreover, we found that the servers picked by a RTT metric performed significantly better on average than those picked by minimizing hop count or AS count, but not as well as our ping-random method. For large files, we found RTT is not a fully reliable predictor of server performance as it often picked servers with transfer times 2-5 times that of the best server. Fig. 3 shows the correlation between average RTT and transfer times for all file sizes in our study. For all clients except UNC the correlations are low. We believe that this is a result of the effect of TCP's congestion control mechanism on file transfer times. Since ICMP PING messages are not subject to this mechanism, the resulting round-trip times do not accurately reflect time required to transfer files.

4.1.1 Round-trip Times

The client at UNC experienced faster transfer times than most of the other clients. We speculate that lower congestion during most transfers at UNC resulted in better correlation for that client.

To understand why good ping times don't necessarily predict good file transfer times, we performed a small *ns*2 (http://www.isi.edu/nsman/ns) simulation to determine the effects of current available bandwidth on RTT. The topology of the experiment is shown in Fig. 5. We studied symmetric FTP traffic over a 1.5 or 2.0 Mbs link. We introduced traffic sources sequentially (one from each end) throughout the experiment. The intermediate links (4–5, 5–6, 6–7 in Fig. 5) had 5 Mbs capacity. We measured the bandwidth in use by monitoring the traffic across higher-capacity links. Although not the focus of this paper, even this small simulation shows that RTTs do give some indication of current available bandwidth. Fig. 6 shows our results. As available bandwidth decreases, roundtrip times tend to increase and the standard deviation of roundtrip times increases. But there is significant overlap in ping times among differing levels of available bandwidth. Thus, a good ping time may be recorded without necessarily indicating sufficient available bandwidth on the network

[1] More extensive results can be found in our technical report [10].

[2] The good performance of RTT for USC 193-set shown in Fig. 2 was not observed for all smaller sets for USC (e.g., Fig. 2 and Fig 9) leading us to believe there was a single server uncommonly predictable by RTT.

Client	10k	30k	100k	250k	500k	750k	1M	Hops
UCSC	0.16	0.20	0.21	0.22	0.21	0.21	0.20	0.07
Purdue	0.18	0.23	0.23	0.23	0.21	0.22	0.22	0.19
UDel	0.14	0.21	0.26	0.30	0.30	0.30	0.29	0.11
UMass	0.21	0.24	0.34	0.39	0.45	0.46	0.45	0.09
UNC	0.39	0.49	0.49	0.50	0.48	0.47	0.45	0.28
USC	0.25	0.34	0.38	0.40	0.40	0.41	0.40	0.23

Figure 3. Correlation coefficients between average ping times and transfer times or hop count.

Client	10k	30k	100k	250k	500k	750k
UCSC	0.31	0.53	0.72	0.80	0.88	0.92
Purdue	0.29	0.36	0.50	0.60	0.69	0.72
UDel	0.36	0.54	0.74	0.84	0.89	0.93
UMass	0.24	0.34	0.55	0.66	0.79	0.85
UNC	0.44	0.66	0.80	0.87	0.91	0.95
USC	0.35	0.55	0.75	0.84	0.90	0.92

Figure 4. Correlation coefficients between 1M file transfer and varying file sizes.

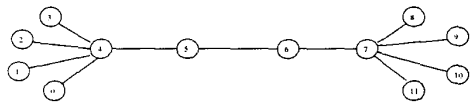

Figure 5. Topology of the ns2 simulation

path. The result also provides intuition into why our ping-random method, discussed subsequently, produces good results: in choosing a subset of servers, it's likely that some of the choices do accurately characterize the network path.

Unlike other clients, Purdue's performance with a RTT selection policy (Fig. 2) was not significantly better than other metrics. We can explain this result: the relative performance of servers was more similar; the chances of picking, even randomly, a server with performance comparable to the best servers was good. At Purdue, the difference in transfer times on average between consecutively ranked servers was strikingly small, as shown in Fig. 7 for both 1 Meg and 100k file sizes. Fig. 7 shows the relative server performance as observed at UCSC for 1 Meg and 100k files, which is representative of all other clients across all file sizes. We conjecture that the similar performance of servers observed at Purdue is likely to have been caused by a common network bottleneck that limited the peak performance of all servers to the client. (We do know that the Purdue client was a slow Intel 80486 machine with limited memory on a constantly congested subnet.)

We expect the varied performance shown in Fig. 7 to be the typical case (if not, server selection is simple). We also expect that resource limitations of the mirror servers themselves will generally not be responsible for observed network performance. We recorded the processor load at 44 servers for one week, and found the loads to be negligi-

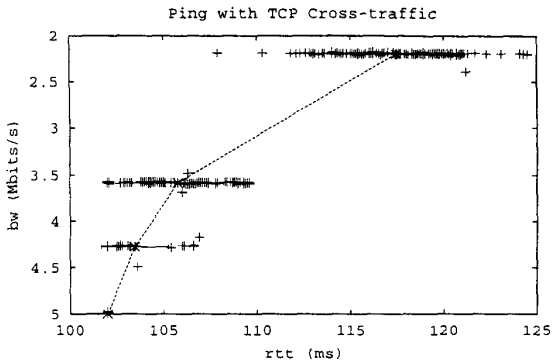

Figure 6. Effects of available bandwidth on ping times.

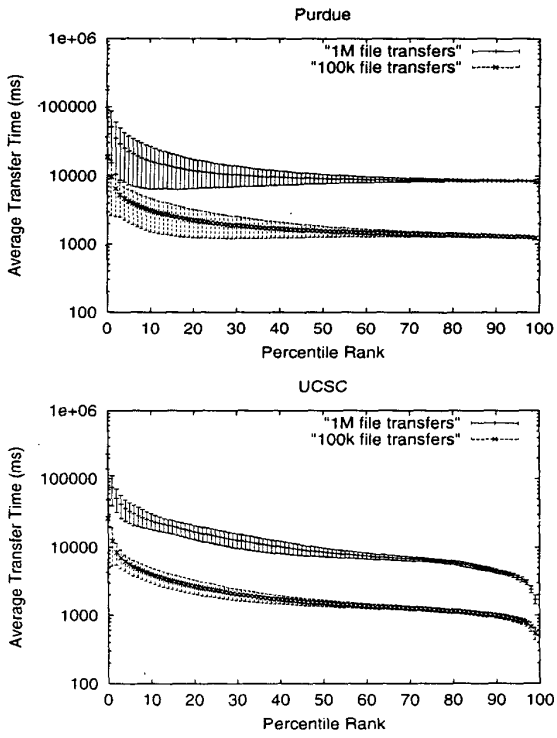

Figure 7. Avg. transfer time of percentile for ranked servers (top: Purdue, 1M and 10k files; bottom: UCSC, 1M and 10K files).

ble and uncorrelated with transfer times; network transfers are not processor-intensive operations. For commercial deployments, mirror servers can be expected to be moderately resourceful.

4.1.2 Small Files as Predictors

Choosing a server based on the transfer times of small files is another metric we analyzed. Fei et al. [7] used a similar metric, but did not report the size of the file used. To use this metric, a client initiates small downloads from all available servers. The server that completes this transfer first is then used for downloading larger files. This method is clearly impractical for larger sets of servers, but we found using a 10k file as a metric for picking the best server for larger file transfers fared well in our experiment. For most clients, on average, a 10k experimental transfer usually picked a server that performed as well as or better than those picked by best ping times.

Not surprisingly, we found larger files were better predictors. We found little correlation between the times required to retrieve small and large files. Fig. 4 shows the correlations of transfer times of 1M files to times of smaller files. These correlations are between files during a single run; i.e., the transfer took place within a minute. We speculate that the poor correlation is the result of TCP slow-start dominating small file transfer. There is a steady trend, and we conjecture that files smaller than 10k will not have an increase in their correlation or resulting performance in predicting server file transfer times.

4.1.3 Ping-Random

In Section 4.1.1 we have shown that selecting a server based on round-trip times as measured by pings does not always produce very good results. We propose a method that uses ping results to pre-select a small set of servers, which we call a *ping set*. This method improves upon ping metrics, takes advantage of their light-weight nature, and exploits the stability of top-ranked servers (discussed in Section 5). Applications can create a ping set by pinging available servers and then selecting the n servers with the best results. From that ping set, some other selection method is used to choose a server for a particular file transfer. We found in our study retaining the subset for 10 days maintained the same average performance, whereas for longer time periods, performance began to drop off. Although the construction of the ping set requires testing between the client and all available servers, the fact that the set is retained amortizes the overall cost.

We examined two techniques for choosing a server from a ping set. With *ping-random*, clients make a random choice, which requires no signalling or delay and produces results that are usually better than other metrics. This method also has an inherent load-balancing property; an interesting parallel can be drawn to work on processor load balancing, e.g., Eager et al. [5]. This method is not based on current network conditions, but it doesn't result in the *oscillation* problem we address in Section 4.3. A large group of clients forming ping sets at the same time is unlikely, and therefore it is improbable that all would choose the same set.

The *ping-ping* method chooses from the ping set by

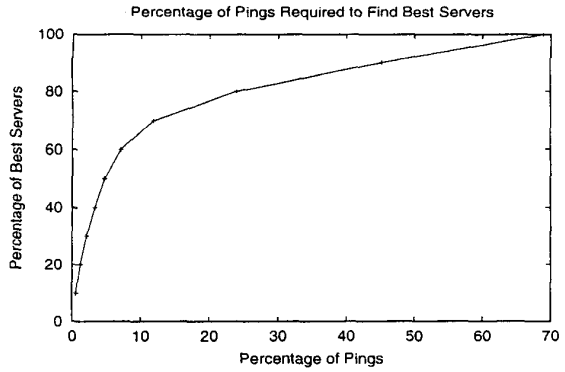

Figure 8. Percentage of servers, ranked by ping, needed to find best servers at UNC.

pinging all its members. This method provided slightly better performance than ping-random (up to one second for the 1M file) for some of the clients in our study, but had no difference for others. However, since pinging five servers is such a light-weight operation, it might prove beneficial in some instances.

The ping-ping metric is greatly influenced by the size of a ping set. The results shown in Fig. 9 are for ping sets of size five, constructed from all 193 servers. Ping sets of size three performed significantly worse. As expected, the chances that a server providing good performance is part of the set increases with the size of the set. We found, on average, 40% of servers that were ever ranked first (and were therefore stable performers, see Section 5) were in the top 20% of ranked pings. Additionally, for four of the clients, 50% of the first-ranked servers were found within the top 12% of ranked pings. The results for UNC, which was a part of the latter group, are given in Fig. 8.

The results for ping sets at UCSC and USC (Fig. 9), including 95% confidence intervals, are representative of all clients. For reference with Figs. 1 and 2, the 10k metric is repeated. We also recorded *ping-best*, the average performance of the best server in the ping set to show an upper bound on the performance possible in any two-step ping set selection method in our study. The ping-best is closest to best of all metrics we tested, e.g., it was usually not more than one half to one second from the best server for a 1M file. Ping-random performs better than selection based on a 10k file transfer with significantly less work required of the client. And, ping-random performs better than a simple RTT metric and also requires less work over time. Ping-ping performed simililary to ping-random for most clients and better only for some. Ping-random Moreover, it may not balance load among servers as well as ping-random.

4.2 Static Selection Methods

In this section, we present our evaluation of router and AS hop count metrics, which are often proposed methods

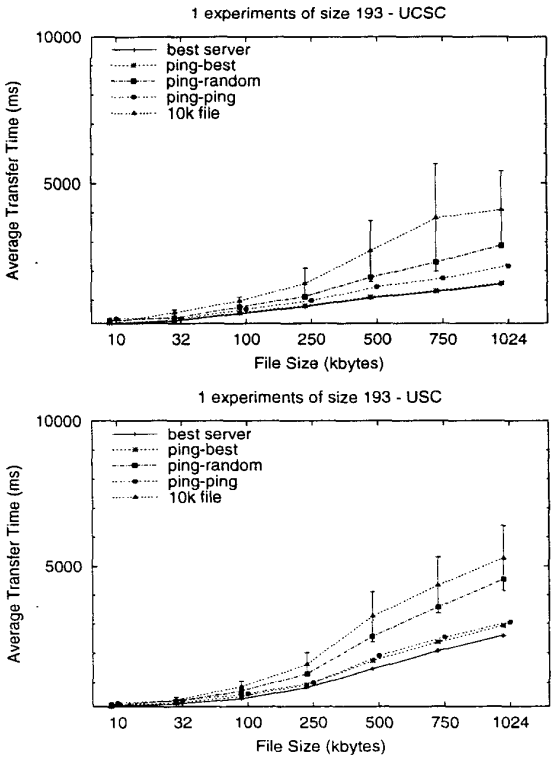

Figure 9. Performance of ping set metrics, $n = 5$ (top: UCSC, 193-set; bottom: USC, 193-set).

of selection. The use of a hop count metric can be either by direct measurement at the application layer [9, 8, 12, 23], or by modifications to unicast [6], multicast [15], or BGP routing [14]. Hop count metrics have been studied at length, but we know of only one other study [19] that has studied the AS metric, which has been recently proposed as part of the GIA protocol. However, that study [19] evaluated only the round trip time to selected servers. We recorded actual transfer times and therefore are able to comment on the performance of such a metric as used by GIA. In our analysis, when multiple servers were at minimum distance, we averaged their observed transfer times as no other information would allow a resolver to make a distinguishing choice in the most generic case.

4.2.1 Hop Counts

The results in Figs. 1–2 suggest that hop-count metrics perform less well than other metrics in predicting the server with the best download time. Often the performance of hop-count metric was similar to picking a server randomly!

Hop count metrics perform poorly because the location of the best server varied greatly while the server selected by minimizing hop count did not. Fig. 13 shows the distri-

bution of hop counts for the best server across all fetches for all file sizes at UCSC (with 95% confidence intervals). The nearest server to UCSC was on average 10 router hops away. This result is representative of all cleints.

Recently, Obraczka and Silva performed a hop count and round-trip time analysis and found the correlation to usually be higher than 50% [19]. For completeness, we computed this correlation as well. The last column of Fig. 3 shows the correlation between RTTs computed as the average of five pings and hop count in our experiment. Our results show much less correlation than observed by that study [19]. Carter and Crovella in 1995 [4] and more recently Sayal et al. [22] found results more similar to our. Almost half of the 601 servers used by Obraczka and Silva were on a different continent for 3 of the clients, and almost 500 for a fourth client, which may explain the difference.

4.2.2 Autonomous System Hops

Selection of mirror servers can also be based on minimizing the number of autonomous systems (ASes) between the client and server. The recently proposed Global Internet Anycast (GIA) [14] protocol uses such a metric. Our experiment provides insight into how well GIA might perform in practice by analyzing a simulation of its server selection policy, which [14] did not provide, nor could we find a similar experiment elsewhere.

Our study supports the claim that selecting servers by minimizing AS hops, like router hops, offers an insufficient level of granularity to distinguish server performance, correlates poorly with network conditions, and provides performance similar to that of random server selection.

GIA searches for mirror servers belonging to an Internet-wide anycast address. Border routers query other Border Gateway Protocol (BGP) peers to resolve the address. Peers answer the query if they have registered a server to the address. Otherwise, the query is passed to other peers in a breadth-first search. The query can travel no further than three hops from the BGP router initiating the query. After collecting replies for a set time, the initiating domain resolves the address to the server that minimizes the number of ASes crossed.

We converted the IP addresses of the routers in the traceroutes we recorded using a tool provided by www.radb.net for "whois" database lookups. We then recorded the transfer times of servers that were closest to each client in terms of AS counts. We could not find a specification of how to break ties among responses, so we averaged the transfer times of all servers that minimized the AS count. We believe this to be fair because the BGP routers would have no extra information by which to distinguish replies. The quickness of BGP routers in responding to requests may have little correlation with the traffic conditions on the path to that server. The BGP router may be delayed with other tasks, but even if the response time of BGP routers is solely limited by the round-trip time, then the poor correlation between round-trip time and transfer time we have observed (see Fig. 3) suggests that picking the earliest response may not be successful.

Figs. 1–2 show results for UCSC, USC, and Purdue. We found that the servers that gave the best transfer times were commonly farther than the closest server and up to 10 AS hops away. Fig. 14 shows a histogram for UMass of the distance in AS hops of the best servers averaged over all file sizes; this histogram is representative of all clients.

GIA suggests sending queries with a maximum hop count of three AS domains, however we found that for all clients, this misses about 80% of all servers. Setting the maximum hop count of search queries in GIA higher to find the best server would create too much query traffic for BGP routers to handle efficiently [14].

4.3 Effects of Selection

In considering server selection techniques it is necessary to examine the effect of a large set of clients using a technique concurrently. Will a particular method result in a large set of clients choosing the same server, where some of them would have received better performance elsewhere? Moreover, will those clients, upon detecting resulting poor performance, change en masse to another server, and then, detecting poor performance at the second server, change again, continuing to *oscillate* among servers?

Static server selection methods such as hop count, AS count and geographical selection do not respond to changing network conditions, and therefore are not subject to the problem of oscillation. However, they may result in a large subset of clients choosing the same server. As we have shown in section 4.2, these metrics are not good predictors of performance; their use by a large set of clients could result in heavy traffic in concentrated areas.

Dynamic selection methods won't result in either problem, as they work by detecting current network conditions. Consider when i clients are currently conducting file transfers from server X. The ith+1 client will only choose X if the selection method indicates that X is the best choice. For good-performing metrics such as ping and ping-random, we can expect the correct choice, and we have shown these metrics usually result in good choices and are responsive to available bandwidth. Furthermore, ping-random evenly distributes the load among the five.

This would not necessarily be the case if a large group of clients began testing at exactly the same time. Cross-traffic notwithstanding, each client would only detect the traffic generated by the testing mechanism (a few ICMP packets). If all clients in the group then chose the same server, the resulting TCP connections could cause enough congestion to result in poor performance. In this scenario, oscillation among servers could occur as well, but only if all clients in the subset re-tested at the same time. Fortunately, it is highly unlikely that a sizeable group of clients would be running the testing protocol at exactly the same time.

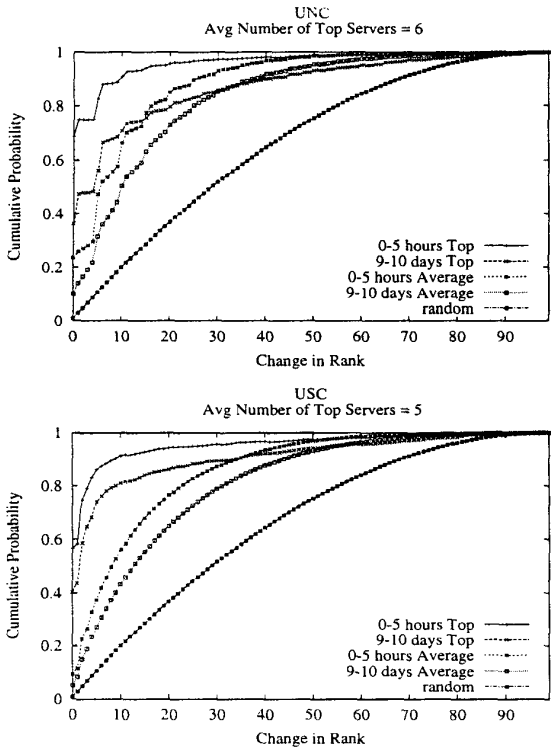

Figure 10. Prob. of rank changes (Top: avg. of 20-subsets for 1M files; bottom: avg. of 50-subsets for 500k files).

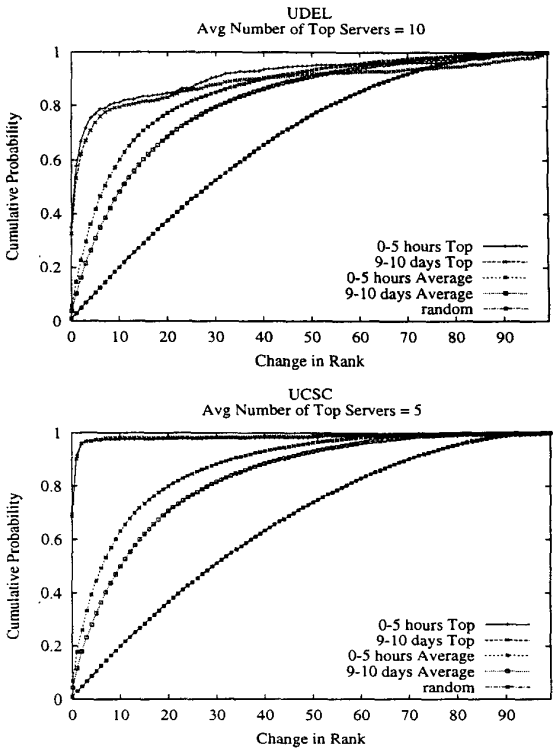

Figure 11. Prob. of rank changes (Top: avg. of 100-subsets for 10k files; bottom: avg. of 193-subset for 250k files).

5 The Stability of Top-ranked Servers

In this section we characterize the performance of the mirror servers in our experiment. We found that the performance of top servers is very stable over time and that the set of top servers is a small fraction of the total number of servers. This supports our findings that choosing a subset of well-performing servers and isolating subsequent testing to that subset results in the best performance for clients.

5.1 Rankings of Top Servers are Stable

We compared servers by creating a ranked list for each run based on sorted transfer times. All our ranks are reported as *percentiles*; a server's percentile is the percentage of servers whose file transfer times were greater than or equal to its own. We examined the rank changes of servers over short and long time scales: 0–5 hours and 9–10 days, respectively. In this section, we compare the rank changes of top servers to the average server.

Fig. 10–11 show the cumulative probability of rank changes of top and average servers, for various clients, subset sizes, and file sizes transfered. Each graph presents the cumulative probability of rank changes for:

- top servers in runs occurring five hours or less apart (labeled "0-5 hours top");
- top servers in runs occurring nine to ten days apart (labeled "9-10 days top");
- all servers in runs occurring five hours or less apart (labeled "0-5 hour average");
- all servers in runs occurring nine to ten days apart (labeled "9-10 days average").

For comparison, each graph shows the cumulative probabilities of differences in ranks when chosen randomly from a uniform distribution. The results presented in this section are typical across all file sizes, n-subset sizes and almost all clients; more extensive results can be found in our technical report [18]. Top servers at UMass and especially Purdue have a smaller proportion of rank changes between zero and ten than top servers at the other clients. We suggest reasons for this difference in Section 4.1.1.

Top servers in our experiment consistently maintain more stable performance than the average of all servers. In the figures, 70–98% of top servers' rank changes in both time scales are between zero and ten. An average server

does not see as large a proportion of small changes; 43–66% of their rank changes in the same time scales are as small. Since our construction of n-subsets creates disjoint sets of servers, the top servers in different n-subsets are actually different servers. The fact that different servers have similar performance characteristics strongly supports the general nature of our result.

Meyers, et al. [17] found that rank changes of servers are independent of the time scale over which they occur. That study accumulated all pairs of server ranks that were ten days *or fewer* apart and plotted the distribution of rank changes that had occurred in that time frame; in contrast, our analysis tracks rank changes that occurred over nine to ten days exactly. They also computed distributions of rank changes for one, two and four hour time scales. They concluded that since all the distributions were very close, rank changes were independent of time scale. We modeled the rank changes of average servers for five hour and ten day time scales using their technique. Although we have not developed precise theoretical measures for judging the closeness of the distributions, we did not observe their result as strongly using our data; i.e., the distributions of the rank changes in the two time scales were clearly separated at all clients.

For completeness, we also used their [17] cumulative time scale technique to model the performance of top servers using our data. Despite using a different methodology, we observed that top servers experienced a larger proportion of small rank changes compared to average servers; a result that mirrors our own finding.

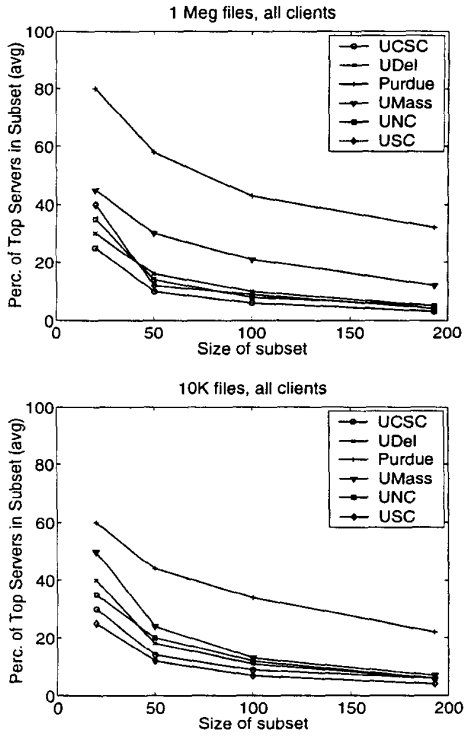

Figure 12. Percentage of top servers in subsets (top: 1M files; bottom: 10k files).

5.2 Transfer Times of Top Servers Are Stable

In our study, we found that very few servers offer the best service to a client over the 41-day period. Therefore, our ping-random selection scheme benefits from narrowing the field to a smaller subset of stable top servers. In this section, we show that such a scheme is inherently scalable with the size of the server population.

Fig. 12 shows the percentage of all servers in 20-, 50-, 100-, and 193-subsets at each client that are in the set of top servers for both the 1M file and the 10k file. Other file sizes produced very similar results.

The figures are strikingly similar. The number of top servers in an n-subset is a small fraction of the size of the set. In fact, as the size of the n-subset increases, the percentage of top servers decreases. Less than ten percent of all servers in the 193-subset are in the set of top servers for all file sizes at four clients. A larger percentage of servers are in the sets of top servers for the clients at Purdue and UMass; we explain this result in the Section 4.1.1. However, the same downward trend in the percentage of top servers with increases in n-subset size witnessed at the other clients also holds for UMass and Purdue.

Two facts give us confidence that the results presented in this paper are representative of top server performance in general. First, there is little overlap between top servers at different clients. The majority of top servers are in the sets of top servers at one or at most two clients. No server in the 50-, 100- and 193-subset was in the set of top servers at all six clients. An average of 3% of top servers in 20-subsets were in the sets of top servers at the six clients. Second, our construction of n-subsets creates disjoint sets of servers; different servers form the set of top servers in different n-subsets. Hence, our results reflect the performance of several individual servers that provide the best performance in their n-subsets.

In comparison, Myers et al. [17] found that on average less than half of all servers need to be considered to achieve good performance most of the time. They defined a server's performance to be *good* if it delivered a document within 110% of the best transfer time for the same document in the same run. In our study, we find that the fraction of servers that needed to achieve best performance all the time is less than ten percent for a large server population. In addition, we find that the percentage of servers that comprise the set of top servers in an n-subset decreases with increases in the size of the n-subset. We have validated these findings across all clients, file sizes, and n-subsets.

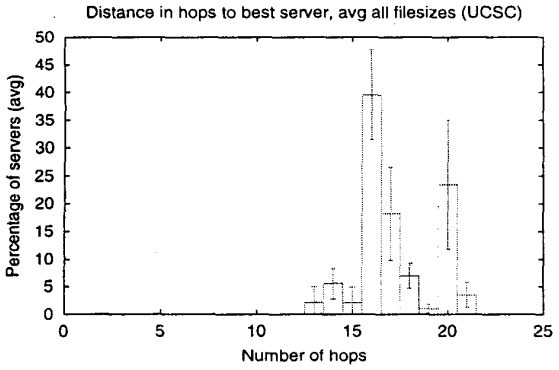

Figure 13. Distribution of distance in router hops to the best server (UCSC).

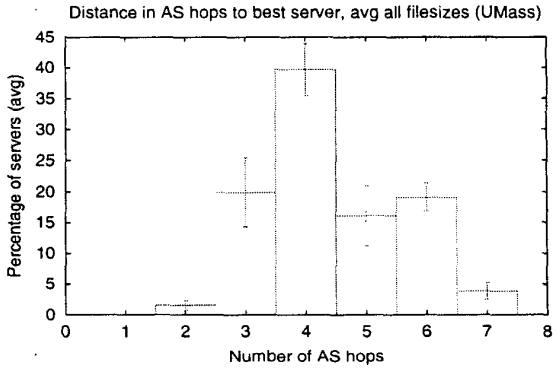

Figure 14. Distribution of the distance to best servers in AS hops for UMass; avg. of all file sizes and all 193 servers.

6 Conclusion

Our results show that our ping-random server selection method performs better than any of the previously-proposed methods we studied. Ping-random takes advantage of the fact that the ranks of top-performing servers remain stable over time, allowing us to eliminate testing over the course of ten days. Moreover, choosing randomly from the ping set helps to balance the load among servers. Since the ping set is formed using round-trip times, a metric that is somewhat sensitive to network traffic, and the set is retained over time, we avoid the problem of clients oscillating among servers. Our experiemnt was extensive and examined server population sizes from 20 to 193.

Acknowledgements

We are grateful to Paul Amer (Univ. of Delaware), Kevin Almeroth (UC Santa Barbara), Mostafa Ammar (Georgia Tech), Jennifer Hou (Ohio State), Kevin Jeffay (Univ. of North Carolina), Tracy Larrabee (UC Santa Cruz), Christos Papadopoulos (Univ. Southern California), and Clay Shields (Purdue University) for providing us with accounts on remote machines. We also thank Christophe Diot and Supratik Bhattacharyya at Sprint ATL and Jim Kurose at UMass Amherst for their insightful comments. A special thank you goes to Scott Swedorski at Tucows for his support and cooperation with our experiment.

References

[1] Akamai. http://www.akamai.com.

[2] S. Bhattacharjee et al. Application Layer Anycasting. In *Proc. IEEE INFOCOM*, pages 1388–96, 1997.

[3] J. Byers et al. Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads. In *Proc. IEEE INFOCOM*, 1999.

[4] R. Carter and M. Crovella. Server Selection Using Dynamic Path Characterization in Wide-Area Networks. In *Proc. IEEE INFOCOM*, 1997.

[5] D. Eager et al. Adaptive load sharing in homogeneous distributed systems. *IEEE Transactions on Software Engineering*, 1986.

[6] R. Engel et al. Using IP Anycast for Load Distribution and Server Location. In *Proc. Global Internet*, November 1998.

[7] Z. Fei et al. A Novel Server Selection Technique for Improving the Response Time of a Replicated Service. In *Proc. IEEE INFOCOM*, 1998.

[8] P. Francis et al. An Architecture for a Global Internet Host Distance Estimation Service. In *IEEE INFOCOM*, March 1999.

[9] J. Guyton and M. Schwartz. Locating Nearby Copies of Replicated Internet Servers. In *Proc. ACM SIGCOMM*, pages 288–298, August 1995.

[10] K. Hanna et al. An Evaluation of Server Selection Metrics for Anycast. UMass Tech Report 01-07, March 2001.

[11] D. Island. http://www.digitalisland.com.

[12] S. Jamin et al. On the Placement of Internet Instrumentation. In *Proc. IEEE INFOCOM 2000*, March 2000.

[13] K. Johnson et al. The Measured Performance of Content Distribution Networks. In *Intl. Web Caching and Content Delivery Workshop*, May 2000.

[14] D. Katabi and J. Wroclawski. A Framework for Scalable Global IP-Anycast (GIA). In *Proc. ACM SIGCOMM*, August 2000.

[15] B. Levine and J. J. Garcia-Luna-Aceves. Improving Internet Multicast with Routing Labels. In *Proc. IEEE Intl. Conference on Network Protocols*, pages 241–50, October 1997.

[16] J. C. Mogul. The case for persistent-connection http. In *Proc. ACM SIGCOMM*, pages 299–313, August 1995.

[17] A. Myers et al. Performance Characteristics of Mirror Servers on the Internet. In *Proc. IEEE INFOCOM*, March 1999.

[18] N. Natarajan et al. A Performance Study of Top Mirror Servers. UMass Tech Report 01-10, January 2001.

[19] K. Obraczka and F. Silva. Network Latency Metrics for Server Proximity. In *Proc. IEEE Globecom*, Dec. 2000.

[20] C. Partridge et al. Host Anycasting Service. IETF RFC 1546, November 1993.

[21] P. Rodriguez et al. Parallel-Access for Mirror Sites in the Internet. In *Proc. IEEE INFOCOM*, March 2000.

[22] M. Sayal et al. Selection Algorithms for Replicated Web Servers. In *Workshop on Internet Server Performance*, June 1998.

[23] Y. Shavitt et al. Computing the Unmeasured: An Algebraic Approach to Internet Mapping. In *IEEE INFOCOM*, April 2001.