An Evaluation of Server Selection Metrics for Anycast*

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

CMPSCI Technical Report TR 01-07

Abstract

Choosing the best-performing server for a particular client from a group of replicated proxies is a difficult task. We offer an empirical evaluation of currently-used and proposed metrics based on traces to 193 commercial proxies. We show that 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, respectively. Other metrics such as round-trip time and tests using small files perform better, but usually select servers that are two to three times worse than the best transfer time possible. We offer two novel techniques that chose a small subset of three to five servers, and isolate testing to that subset for ten days. We show that our techniques perform better than any of the other metrics we studied.

1 Introduction

Sites providing web content, multimedia streaming, networked gaming, or other Internet services commonly need to scale to large client bases. One solution is to increase the processing or bandwidth resources of the site's server. Another solution is to replicate the server in numerous locations in the Internet. This technique provides a number of advantages over a centralized approach and is often adopted in practice.

Presenting numerous mirror servers to a client results in the difficult problem of finding the server that will perform best. *Anycast* protocols allow a client to transparently discover and choose a server of a set of replicated servers [18] — hopefully the best server. Proposals for implementing an anycast service range from network- to application-layer approaches and are typically based on specific *metrics*, such as hop count or round-trip time. Anycast-like techniques have also been proposed for use in Internet distance maps.

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 anycast services are commonly based on DNS modifications [1, 9]. 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 [11].

We address the following fundamental question: *what metric for server selection results in the best performance for clients?* To answer this question we evaluate, through experimentation and analysis, four approaches to server selection: minimizing router counts, autonomous system (AS) counts, round-trip times, or transfer times of small test files. We also evaluate a novel technique of our own design called *pre-selection*, which is a two-step process for server selection. The first step isolates a subset of 3–5 well-performing servers for a period of 10 days; the second step selects among that subset for downloads during the 10-day period. Our results indicate that:

- network-layer metrics perform very poorly as predictors of the best server from which to retrieve files;
- purely receiver-driven testing is sufficient to find servers that deliver very good performance; large infrastructure deployments for testing are not needed;

*This work is supported in part by a grant from Sprint Advanced Technology Labs and NSF grant EIA-0080119.

- our techniques of *pre-selection* can select servers with very good performance and are efficient with large server populations; metrics based on router hops, AS hops, round-trip times, and small files perform worse.

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 results. Section 5 offers our concluding remarks.

2 Background

Replicated server networks have three major components: the original sources of content; distributed proxy servers carrying replicated content; and end-users desiring fast access to that content. In order to deliver the lower latency made possible by such a network, anycast protocols allow users to discover and pick the best of the proxy servers [18].

Anycast protocols can be designed to meet a number of objectives. Our goal is to determine which metric allows a client to choose the best-performing server from a set of servers that can provide the needed service; in this case, transferring the requested document to the client. We do not focus on techniques for load-balancing or resource discovery in this paper. We concentrate instead on situations where network conditions are the constraints and where all mirror sites are known and are exact replicas. Load-balancing can be a priority for applications where lack of resources at the server produce a bottleneck, but here we assume servers are well-provisioned and performance is most affected by network conditions.

2.1 Previous Work

Previous work on anycast services may be taxonomized into approaches working at the network and application layers. Most generally, proposed network-level anycasting protocols [5, 13] are forced to ignore dynamic network conditions and instead focus on discovering the best server by minimizing network distances. Application-level proposals [6, 2] tend to consider combinations of network and server performance in selecting the best server. However, application-level proposals often require substantial application-level signaling to configure routing and are not easily able to aggregate that signaling across domains. Each domain must perform its own set of tests to determine anycast routes. Such testing can be taxing to popular anycast servers as well as intervening network links.

One network-layer approach to anycast attempts to minimize the number of routers traversed [5, 13]. This approach utilizes existing unicast routing tables to resolve multiple servers sharing the same anycast address. This is unscalable as the route toward each global anycast address would have to be stored at each router. GIA [12], a recently proposed network-layer scheme, attempts to minimize the number of domains crossed (i.e., the number of BGP peers) between client and server. As we show in this paper, these approaches are not good predictors of file transfer times between a client and server. Details of GIA are discussed in Section 4.

An application-layer approach to anycast has been proposed [2] that involves a resolver situated within a domain. Clients query the resolver with an *Anycast Domain Name* that identifies the desired service and the resolver chooses a server and provides that choice to the client. The resolver is configured to make server choices based on metrics, policies, or some combination of the two, or through arbitrary means. We find that architecture to be flexible and compatible with our work, as we evaluate a variety of metrics that could be utilized in the context of that approach.

Also related are techniques for creating Internet distance maps [10, 7], which allow an arbitrary machine to determine its distance to any other point on the Internet. The maps are created with a distributed set of *tracer* machines. In this paper, we evaluate the metrics that have been used to build IDMaps, hop count and ping times, and we believe the contributions of this paper can be applied to distance map techniques for more accurate maps.

Previous work [14, 16] has also shown that there may exist consistency in server rankings even when exact performance may change between sessions.

There have been several past studies on determining appropriate server selection metrics [4, 20, 17, 6, 2]. Some of these studies produced results that differ from ours. We comment on these results where appropriate in Section 4 of this paper. In contrast to those studies, our measurement study uses six clients, increases the number of servers by an order of magnitude to almost 200, has longer measurement period of 41 days, uses larger file sizes up to 1 Meg, compares many selection metrics at once, and for the first time considers the two-step pre-selection process for server selection. Additionally, our study uses actual transfer times for files of varying sizes where some have used estimates or not considered transfer times at all.

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 1: Client Sites and Data Characteristics

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 1). Each of these clients interacted with 193 servers that are part of the tucows.com network of web mirrors. The tucows servers were located throughout the U.S. and Canada.

Data was collected via a script that ran continuously at each client over the course of 41 days, from September 30 – November 9, 2000. The script was implemented in Tcl/Expect, and made use of standard system utilities. At each *run* of the script, the client collected several types of data regarding the characteristics of the network path between the client and each server:

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

We then explored several different ways in which such data could be used in server selection. Ping results were used to evaluate the performance of a server chosen by minimizing round-trip times. We computed both router hop counts and autonomous system counts from the traceroutes. We used these counts to characterize the performance of selection schemes that minimize router hop counts or autonomous system (AS) counts. Domain-hops were calculated by querying a whois database for an AS number for each IP address in the trace.

At each client there were occasional instances in which a run was aborted due to client failure or disconnection from the network. These aborted runs were relatively rare (Table 1). The disparity among clients in the numbers of runs performed 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.

Due to the extended time it took to contact 193 servers from each client, network conditions may have changed from the beginning to the end of a run. On the other hand, accessing even a small number of servers in parallel would have the file transfers competing with each other for bandwidth. Therefore, we cannot account for short-term changes in network conditions in our analysis. However, we observed that our results remain stable over periods of ten days, so we believe short-term effects are not significant.

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.

It is possible that one or more servers in our experiment employed DNS rotation techniques to map a textual address to more than one server. We don't know if this is the case, but since our study focuses on performance as experienced by a client, this would not affect our results.

4 Server Selection Metrics

Recall that our goal is to find the metric that is best able to predict which server will provide the best performance. 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 given 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

- hop counts,
- AS counts,

- round-trip times,
- transfer times of a 10k file,
- random selection,
- *ping set pre-selection*,
- *transfer set pre-selection*

Transfer sets and *pings sets* are two techniques we designed that base server selection on a two-step process: first, they pre-select a subset of 3–5 well-performing servers; second, they select among the servers in that subset for downloads during a 10-day period. Transfer sets isolate subsets based on ping times and then 250k file transfers; ping sets isolate subsets based on ping times only. We discuss the details of the pre-selection techniques in Section 4.5.

Previous work regarding the performance of mirror servers [14] has shown that the past performance of a server is often a reliable predictor of the future performance of that server. Our related work has shown that servers with the best transfer times are even more stable than the average server [16]. This observation lead us to design the pre-selection techniques.

Our study indicates that our pre-selection techniques perform best and require the least work of all metrics. Network-layer metrics — minimizing hop count and AS count — perform poorly as predictors of file transfer times, often as poorly as random selection. RTT times are also poorly-correlated to transfer times and are not accurate predictors. Additionally, we found transfer times of small files to be poor predictors of transfer times of larger files.

Our experiment offers no results on the affect of a large set of clients all following the same technique. None of the techniques prevent many clients from choosing the same server. However, we conjecture that for techniques that are reactive to network conditions, it is unlikely that all clients from a large client base would choose the same server; a substantial group of clients gravitating toward a single server would affect the tests of later clients.

Our results are presented over a variety of file sizes. For smaller file sizes, the metrics did not distinguish themselves greatly. Additionally, small files make it difficult to justify taking time to test servers. Web pages often contain many small files, but their transfer is commonly aggregated with persistent and pipelined HTTP connections, and therefore it is likely that our results for larger file sizes are more widely applicable.

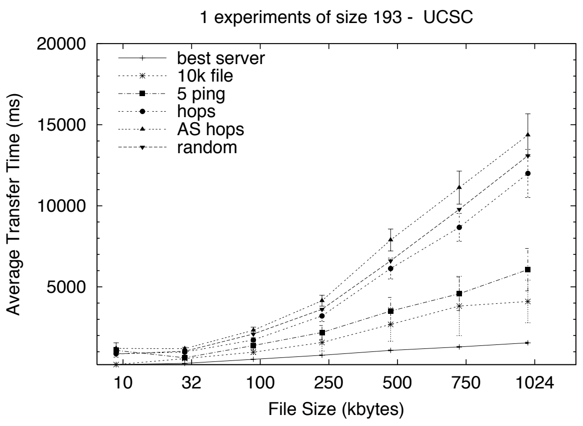

Figure 1: Performance of server selection metrics (UCSC, 193-set).

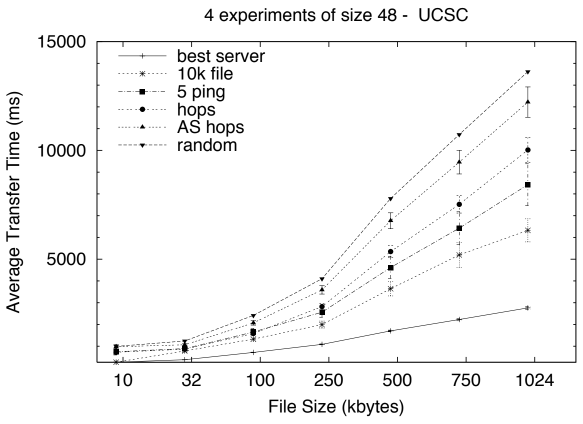

Figure 2: Performance of server selection metrics (UCSC, 48-subset).

4.1 Hop Counts

Server selection based on minimizing the number of router hops between a client and server is used by many proposed server selection services. The use of a hop count metric can be either by direct measurement [8, 7, 10, 21] at the application layer or by modifications to unicast or multicast routing [5, 13]. 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.

Our results for some clients are shown in Figs. 1–5 for a variety of metrics at once, 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 in-

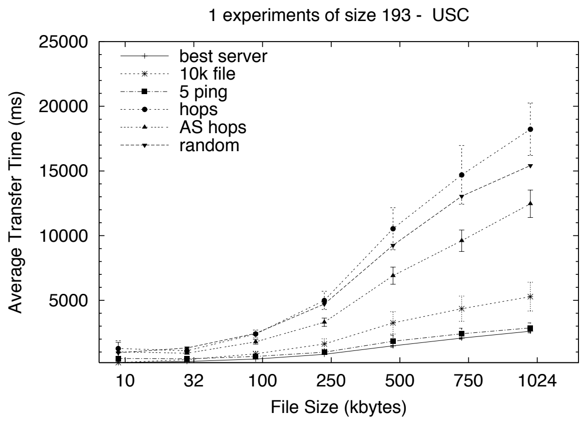

Figure 3: Performance of server selection metrics (USC, 193-set).

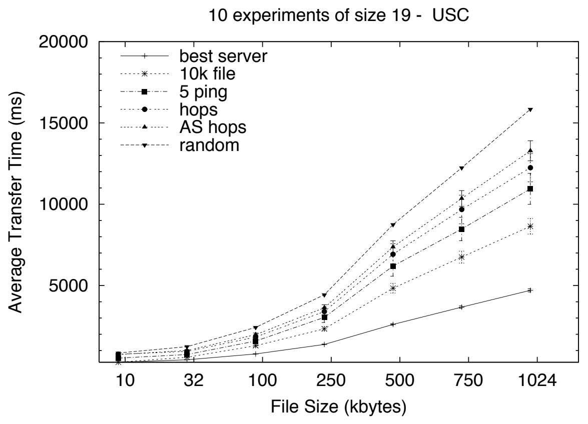

Figure 4: Performance of server selection metrics (USC, 19-subset).

terval of the averages. The number of run completed at each client ranged from 235–328 for the 41-day period of data collection, except for U. Mass where only 131 runs were completed.

Our results suggest that hop-count metrics perform less well than other metrics, such as round-trip times, in predicting the server with the best download time. Often the performance of hop-count based server selection was similar to picking a server at random!

Figs. 6 and 7 offer explanation as to why hop count performs poorly as a server selection method. The figures show the distribution of hop counts for the best server across all fetches for all file sizes at UCSC and USC; error bars represent 95% confidence intervals of the data. Though not marked on the graph, the nearest server to UCSC was on average 10 router hops away with almost no variation, and the nearest server to USC was 11 hops away with almost no variation. The location of the best server varied quite a

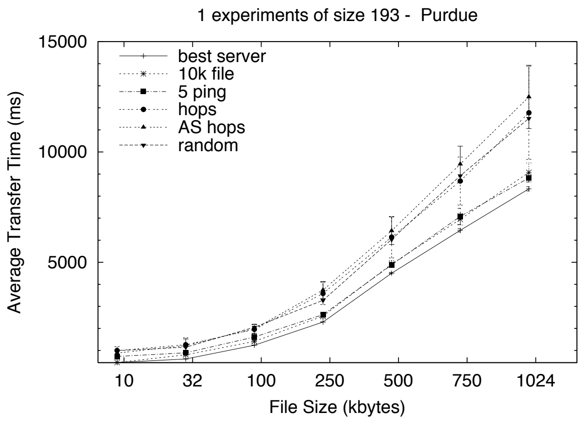

Figure 5: Performance of server selection metrics (Purdue, 193-set).

bit while the server selected by minimizing hop count did not.

Purdue's performance with a minimal hop count policy (Fig. 5) relative to the best server was significantly better than UCSC and USC, as well as that of the other clients. Fortunately, this gave us insight into the general performance of hop counts as a metric.

At Purdue, the difference in transfer times on average between consecutively ranked servers was strikingly small, as shown in Figure 8 for both 1 Meg and 100k file sizes. Figure 9 shows the relative server performance as observed at UCSC for 1 Meg and 100k files, which is representative of all clients except Purdue across all file sizes.

We believe that for Purdue, static measures such as hop count and AS count (discussed subsequently) performed better because the relative performance of servers was more similar: the chances of picking a server with performance comparable to the best servers was good. We see that randomly choosing a server also performed better at Purdue, relative to the best server, again due to the closer relative performance of servers. In other words, hop count is only slightly better than picking a server at random.

Many factors in a network will cause file transfer times and TCP performance to vary over different servers, including packet loss, available bandwidth, and round-trip time. We conjecture that the similar performance of servers observed at Purdue was likely to be caused by a common network bottleneck limiting the peak performance of all servers to a client. (We do know that the Purdue client was on a constantly congested subnet and running on a slow Intel 80486 machine with limited memory.) The more

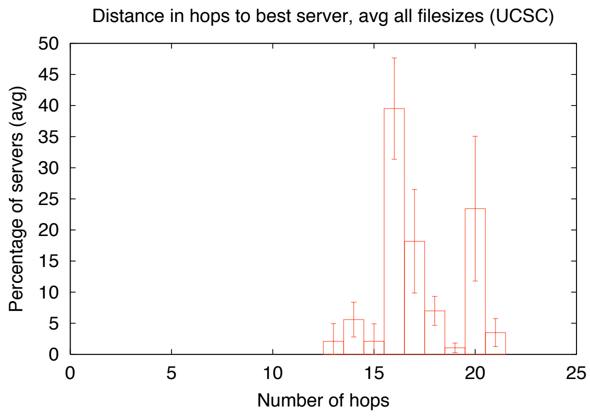

Figure 6: Distribution of distance in routers hops to the best server for UCSC.

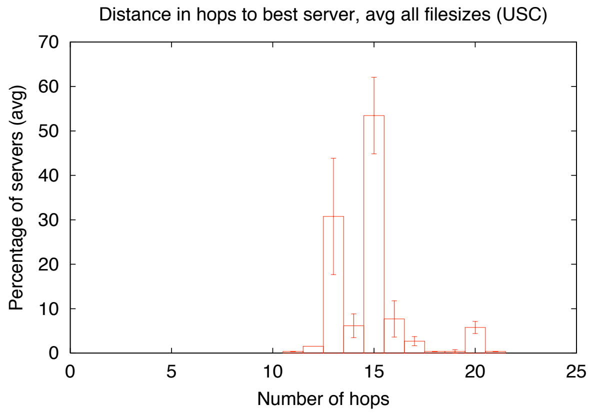

Figure 7: Distribution of distance in routers hops to best server for USC.

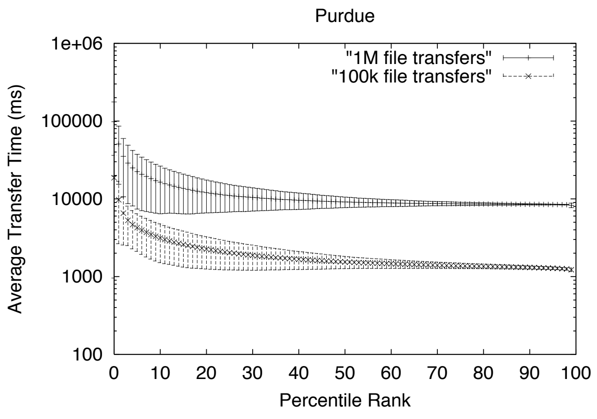

Figure 8: Average transfer time of percentile for ranked servers (Purdue, 1M and 10k files).

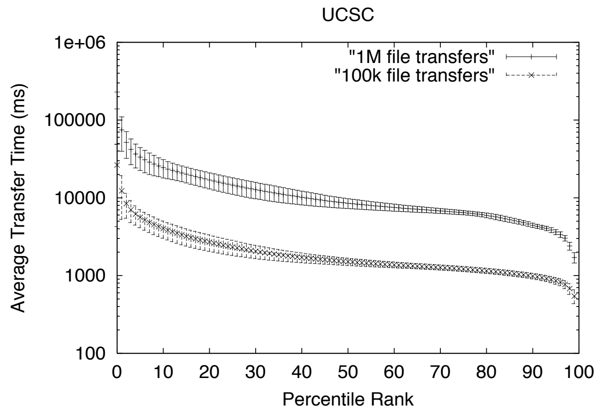

Figure 9: Avg transfer time of percentile for ranked servers (UCSC, 1M and 10K files).

common case in our experiment were clients that had varied relative server performance.

The shape of the relative performance curves shown in Fig. 9 for servers was extremely similar for all clients except Purdue. We expect such varied performance to be typical. 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 negligible and not correlated with transfer times; network transfers are not processor-intensive operations. For commercial deployments, mirror servers can be expected to be moderately resourceful.

We expect clients in the Internet to have varied relative server performance, and therefore we believe static network-based measures to be generally of less use and poor predictors of file transfer performance.

Recently, Obraczka and Silva performed a hop count and round-trip time analysis and found the correlation to usually be higher than 50% [17]. For completeness, we computed this correlation as well. The last column of Figure 12 shows correlation coefficients between round-trip times computed as the average of five pings and hop count in our experiment. Our results show much less correlation than observed by Obraczka and Silva. Carter and Crovella examined the relationship between hop count and round-trip times in 1995 and found results more similar to ours [4]. It may be that correlation between hop count and round-trip time is important for streaming media applications, but it is likely that the correlation between hop count and transfer times is more applicable to supporting file transfer applications; but more importantly, we believe the file transfer times resulting from these policies (Figs. 1–5) are more telling and reliable than correlation measures.

4.2 Autonomous System Hops

Selection of mirror servers can also be based on minimizing the number of autonomous systems (AS) crossed between the client and the server. Katabi and Wroclawski have proposed the Global Internet Anycast (GIA) [12] protocol. GIA searches for mirror servers belonging to an Internet-wide anycast address. Routers resolve the unicast destination of the address by distinguishing unpopular and popular anycast addresses. For unpopular routes, packets are routed to a default unicast address encoded in a portion of a 32-bit anycast address. For popular routes, border routers of a domain query Border Gateway Protocol (BGP) peers for resolution of the address. If the peers have a mirror server registered to that anycast address, they will answer the query. Otherwise, the query is passed to another set of peers, as 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 interval of time, the initiating domain resolves the anycast route with the mirror server that minimizes the number of ASes crossed.

Our experiment provides insight into how well GIA might perform in practice by analyzing a simulation of its server selection policy. We conducted traceroutes between a client and the servers in each run of our script. We converted the IP addresses of the routers in the traces 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.

In this experiment, there were often several servers that were of the same minimum distance. GIA collects multiple responses during a set time interval, but we could not find a specification of how to break ties so we averaged the transfer times of all such possible servers. 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 Figure 12) suggests that picking the earliest response may not be successful.

Figs. 1–5 show results for UCSC, USC, and Purdue. Minimizing AS hops was not a good predictor of servers with low transfer times in our study. Similar to our results for router hop counts, 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. Figure 10 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.

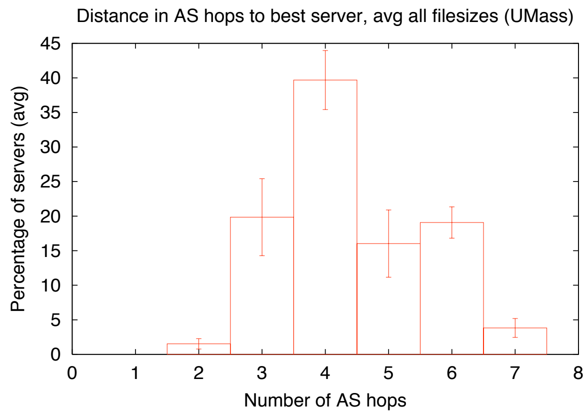

Figure 10: Distribution of the distance to best servers in AS hops for UMass, averaged for all file sizes and for all 193 servers.

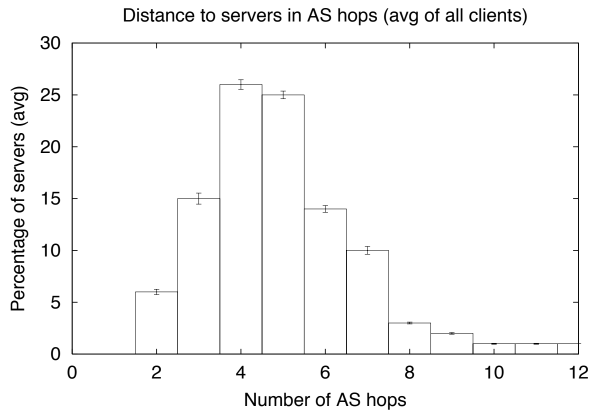

Figure 11: Distribution of all 193 servers from all clients in AS hops.

GIA suggests sending queries with a maximum hop count of three AS domains, however, as shown Figure 11, for the UMass client 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 [12].

Overall, our study supports the claim that selecting servers by minimizing AS hops, like router hops, offers an insufficient level of granularity to distinguish the performance of servers, does not correlate well with network conditions, and provides performance similar to that of random server selection.

Client	10115	31800	107023	236694	524934	765736	1007102	Hop count
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 12: Correlation coefficients between average ping times and transfer times for varying file sizes.

4.3 Round-trip Times

Determining the round-trip time (RTT) to a remote server using an ICMP echo request is normally a simple and quick operation compared to determining hop count or AS counts. Round-trip times, also called 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. For example, Napster, a popular peer-to-peer system [15] measures round-trip times to aid users in selecting the peer from which to retrieve a file.

In our experiment, clients performed five pings in immediate succession to servers. We found that this RTT metric is better-correlated to actual file transfer duration than is hop count or AS count. Moreover, we found that the servers picked by a RTT metric performed better on average than those picked by minimizing hop count or AS count. Unfortunately, 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 the transfer time of the best server except for Purdue where all metrics fared well, as discussed previously. The performance of the RTT metric can be observed in Figs 1–5[1]. Figure 12 shows correlation coefficients between average RTT and transfer times for all file sizes in our study. For all clients except UNC the correlations are quite low. We believe that this is a result of the effect of TCP's congestion control mechanism on file transfer times. Since ICMP messages such as those used for ping are not subject to this mechanism, the resulting round-trip times do not accurately reflect time required to transfer files. 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.

[1]The good performance of RTT for USC 193-set shown in Fig. 3 was not observed for all smaller sets for USC (e.g., Fig 4), leading us to believe there was a single server uncommonly predictable by RTT.

4.4 Small Files as Predictors

Another server selection method is to choose a server based on the time it takes to retrieve a small file. We are unaware of another study that has considered the performance of this metric. To use this metric, an entity initiates small downloads from all available servers. The server that completes this transfer first is then used for downloading larger files. For example, for choosing between a small group of five servers, a client might first try a 2k transfer from each. 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. As we show in the next section, if the set of servers can be trimmed by the client through the use of other metrics, a small file transfer is the best metric and a reasonable choice for choosing a server from that small subset.

Not surprisingly, we found larger files were better predictors. In our experiment we found little correlation between the time required to retrieve small files and that required to retrieve large files. Figure 13 shows the correlations of transfer times of smaller file sizes to the transfer times of 1M files. These correlations are between the transfer times of files during the same run of the script, i.e., the transfer took place within a minute or two. We speculate that the poor correlation of small file sizes to larger file sizes is the result of TCP slow-start dominating the transfer times of small files. The performance observed during this phase does not accurately predict performance during the rest of a 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.

Client	10115	31800	107023	236694	524934	765736
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 13: Correlation coefficients between 1M file transfer and varying file sizes.

4.5 Pre-selection Methods

In this section, we propose the use of a *pre-selection* scheme that divides the testing phase of server selection into two parts. First, an entity chooses a subset of potential servers from the full set of available servers, possibly by a somewhat imprecise, lightweight means. The subset remains constant for a length of time; in our experiment we re-selected subsets every ten days. For each download, the second step is taken: a server is chosen from the subset through a different testing method. We believe the success of this method is derived from the good stability of the best servers, which we have observed for this same experiment and have reported elsewhere [16]; poor performing servers did not exhibit this stability. The first step in the pre-selection scheme aims to identify servers which perform better and therefore are likely to have better and more stable performance over long periods of time; the second step aims to pick the best of those candidates for each download.

Our previous work [16] demonstrated the performance achievable by identifying top servers and relying on their stability, showing that past performance of a server is a reliable metric for future performance. The server with the smallest 250k download time was kept as a choice for 29 days and gave the client near-best performance on average for the 29 day period. That method involved lengthy testing — a 250k file transfer from each client, taking on average 11 minutes to complete — its feasibility may depend on several factors, including the number of available servers to be queried, and whether testing is performed at each client or by a resolver shared by many clients.

In this section, we show that pre-selection methods show promise of performance similar to the heavy-weight selection metric [16] but with testing that is significantly less costly. We outline two pre-selection schemes, *ping sets* and *transfer sets*, both of which culminate in server selection by a *parallel transfer* method. Several papers have proposed improving file transfer performance by retrieving replicated content in parallel from several servers [3, 19]. In these schemes clients contact a set of servers hosting the desired content and retrieve different portions of the file from each. In the case of TCP file transfers, we believe that parallel transfer methods result in a client receiving an unfair share of bandwidth. This is due to the additive increase, multiplicative decrease (AIMD) congestion control algorithm employed by TCP. As congestion on a link grows, AIMD ensures that all flows receive roughly equal bandwidth on that link. When more than one flow to a client travels across a link, the client receives multiple shares of the bandwidth. In one parallel transfer scheme [19], the authors assume that the servers in use do not share bottleneck links. We believe that this assumption is unrealistic in practice. However, we propose that parallel transfer methods be applied to selecting a server for file transfers.

In our schemes, we assume an algorithm that begins a parallel transfer, monitoring the performance of each server. Connections to all but the best-performing server are quickly dropped and the download continues from the chosen server. We do not offer a specific method of parallel server selection here, but we do offer an evaluation of the best and worst performance such an algorithm can achieve, discounting practical overhead. We present the following results for our pre-selection methods: *ping set best*, *transfer set best*, *worst*, and *naive*. Our ping set best and transfer set best represent results that assume a parallel method that chooses perfectly and always finds the best server from the chosen subset. Worst represents the results of a random choice from the subset with 1% of outliers removed, illustrating the worst a parallel selection algorithm could perform with no work. The naive results represent a parallel algorithm that chooses from the subset based on the results of the first 10k transferred. (We did not subtract the time that would be saved by downloading the first 10k of the file in presenting the resulting transfer time; the time reported is the total transfer time of the entire file after selection of the server.)

We compare these three results with the result of picking the best server, as well as the results of choosing the best performer of a 10k file transfer from the complete set (the same *best* and *10k file* metrics used for Figs. 1–5). Due to space limitations we are unable to present all of our results. The results that we do present are generally representative of other clients and different sizes of server subsets. We have observed some variation over the full set of results; we present variations when they are significantly anomalous or run counter to our general claims.

As shown in subsection 4.5.2 the best performance that can be achieved by pre-selection methods is quite good. For most of the clients in our experiment, the worst that can be expected — random selection from the subset — is still better than the performance achieved using RTT metrics (discussed in Section 4.3). That is, for most clients, if a parallel selection algorithm can make a better-than-random choice even some of the time, that server will, on average, perform better than servers chosen by a round-trip time metric or 10k file transfer metric while requiring considerably less work. As shown by the ping-set best and transfer-set best results, a sophisticated metric could produce results that are close to selecting the best server.

4.5.1 Ping Sets

In Section 4.3 we have shown that selecting a server based on round-trip times as measured by pings, although better than network-layer approaches, cannot be relied upon to produce desirable 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, while taking advantage of their light-weight nature. In order to create a ping set an application pings all available servers and selects the n servers with the best results. From that ping set, a parallel algorithm is used each download to select the current best server. In our study, we found retaining the subset for 10 days maintained the same average performance, whereas for longer time periods, performance began to drop off.

We found that the size of a ping set greatly influences the performance of the metric. The ping sets for the results shown in Figures 15 and 16 were of size five, constructed from the entire set of 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 as the size of the set increases. In our experiment we found that, on average, 40% of servers that were ever ranked first could be found within the top 20%

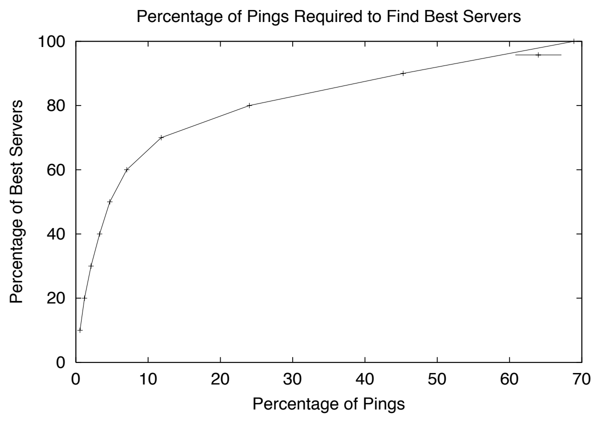

Figure 14: Percentage of ranked pings needed to find best servers at UNC.

of ranked pings. Additionally, for four of the clients, 50% of the top 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 Figure 14.

The results for ping sets at UCSC and USC (Figs. 15 and 16), which include 95% confidence intervals, are representative of all clients. For reference with Figs. 1 and 3, the 10k metric is repeated. For a variety of file sizes, the ping-set best metric is closest to best of all metrics we tested, usually not more than one half to one second from the best server for a 1M file. In the worst case (a random choice of server from the ping set for each download), ping set techniques perform better than selection based on a 10k file transfer with significantly less work required of the client. Additionally, random ping set selection performs better than a RTT metric also for less work. A naive selection among the ping set performs moderately and consistently better than worse case selection, but not as well as the best case selection for the ping set. We believe future work can find a selection method for ping sets that is closer to best than the naive approach.

4.5.2 Transfer Sets

Implementation of a parallel selection algorithm is beyond the scope of this paper. Therefore, we do not know the number of servers such an algorithm can efficiently employ. We offer another pre-selection method that can reduce the number of servers in the resulting set, while providing performance comparable to that of ping sets.

In order to reduce the number of servers used in the parallel selection step, we expand the pre-selection phase to include a second, refining step. A *transfer*

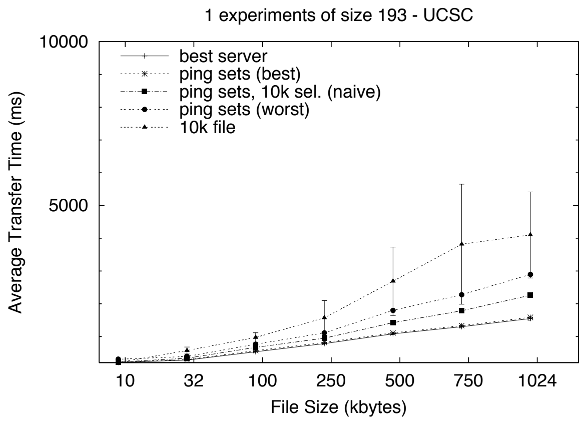

Figure 15: Performance of ping set metrics, $n = 5$ (UCSC, 193-set).

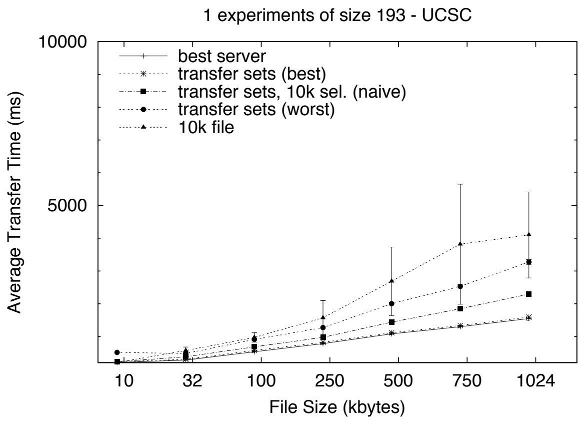

Figure 17: Performance of transfer sets $n = 5, m = 3$ (UCSC. 193-set).

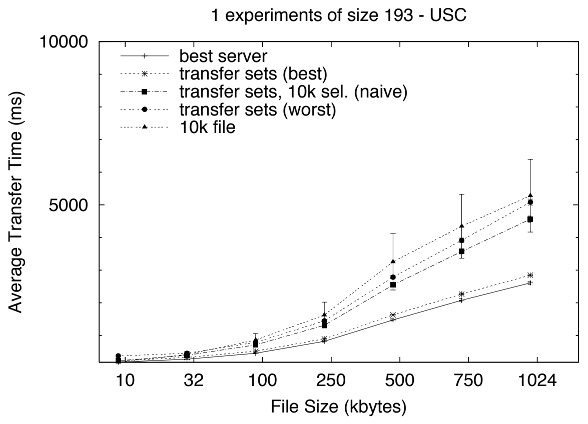

Figure 16: Performance of ping set metrics, $n = 5$ (USC, 193-set).

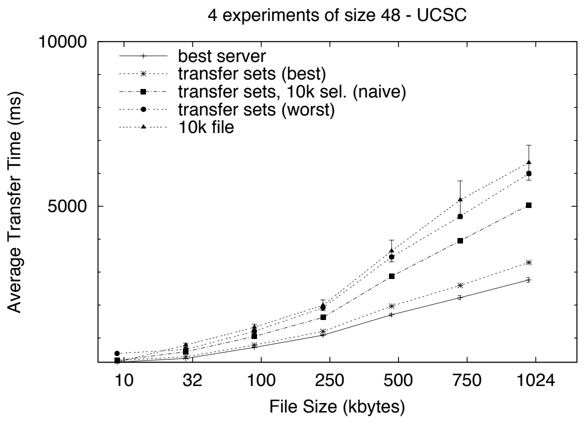

Figure 18: Performance of transfer sets $n = 5, m = 3$ (UCSC. 48-subset).

set is created from a ping set by conducting experimental file transfers to determine the $m < n$ best servers in the set, where n is the size of the ping set. This transfer set is then used by the parallel algorithm to choose the best server. We observed that, on average, transfer sets perform as well as ping sets, while requiring less work during the parallel selection phase.

To simulate transfer sets we used ping results from the full set of servers for our first pre-selection step. For the second pre-selection step we used file transfer results from 250k files. We conducted experiments using both the 1M and 250k files for this second step and found little discernible performance difference. This result corroborates our recent work [16] which has shown that transfer times of 250k files accurately predict servers with good transfer times for all file sizes in the same experiment.

We compare these results to the best result possible from the complete server set, as well as to the result obtained by choosing the server with the shortest transfer time of a 10k file. Figures 17, 18 and 19 show the three transfer set results for the clients at UCSC and USC. In these experiments, $n = 5$, $m = 3$; these values were chosen to show that transfer sets had performance equal to ping sets while using a smaller subset during transfers. All of the transfer set methods perform better than the 10k file method, which, as shown in Figures 1 and 3, is generally the best-performing of the simple methods. The transfer set method using naive parallel selection performs as well as or better than a random choice from the transfer set and produces results similar to or better than the 10k file transfer method. Moreover, we believe that it provides a good approximation of what an actual parallel algorithm is likely to achieve. It clearly requires less work, as construction of the transfer sets occurs only once every ten days. This construction

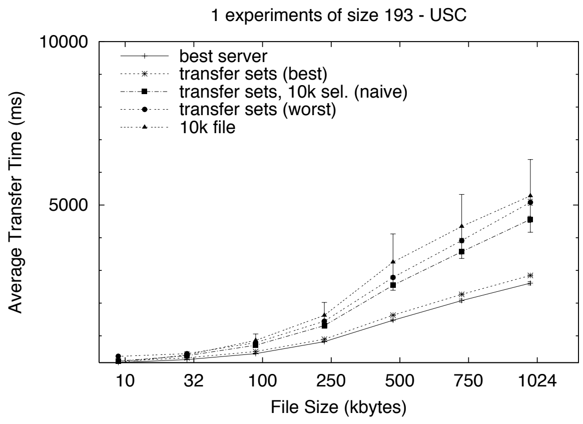

Figure 19: Performance of transfer sets $n = 5, m = 3$ (USC. 193-set).

involves pinging all available servers and transferring 250k files from a small subset of them. In contrast, the 10k file transfer method requires transferring files from all of the available servers at each file request.

5 Conclusion

Our results show the best performance possible from our pre-selection methods is close to that of choosing the best possible server, while the worst these methods can perform is better than any of the metrics we evaluated. Naive approaches perform consistently better than worst, though not as well as the best choice possible. Metrics based on minimizing router hop count and autonomous system count often choose servers that perform four to six times worse than choosing the best possible server. Other metrics, such as round-trip time as measured by ping and transfer times of small files deliver performance on average two to three times worse than choosing the best possible server.

Our ping-set and transfer-set anycast techniques achieve very good performance, but do not require a pre-deployed infrastructure. Our evaluation of these pre-selection techniques shows that anycast protocols based on client testing can perform well without the overhead of constantly testing a large set of servers.

6 Acknowledgements

We are grateful to Paul Amer at the Univ. of Delaware, Kevin Almeroth at UC Santa Barbara, Mostafa Ammar at Georgia Tech, Jennifer Hou at Ohio State, Kevin Jeffay at Univ. of North Carolina, Tracy Larrabee at UC Santa Cruz, Christos Papadopoulos at Univ. Southern California, and Clay Shields at 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, M.H. Ammar, E.W. Zegura, V. Shah, and Z. Fei. Application Layer Anycasting. In *Proc. of IEEE INFOCOM'97*, pages 1388–96, 1997.

[3] J. Byers, M. Luby, and M. Mitzenmacher. Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads. In *Proc. IEEE INFOCOM'99*, 1999.

[4] R.L. Carter and M.E. Crovella. Server Selection Using Dynamic Path Characterization in Wide-Area Networks. In *Proceedings of 16th IEEE INFOCOM'97*, 1997.

[5] R. Engel, V. Peris, E. Basturk, V. Peris, and D. Saha. Using IP Anycast for Load Distribution and Server Location. In *Proc. Third Global Internet Mini-Conference in conjunction with Globecom '98*, November 1998.

[6] Z. Fei, S. Bhattacharjee, E. Zegura, and M.H. Ammar. A Novel Server Selection Technique for Improving the Response Time of a Replicated Service. In *Proc. IEEE INFOCOM'98*, 1998.

[7] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. Gryniewicz, and Y. Jin. An Architecture for a Global Internet Host Distance Estimation Service. In *Proc. IEEE INFOCOM 1999*, March 1999.

[8] J. Guyton and M. Schwartz. Locating Nearby Copies of Replicated Internet Servers. In *Proceedings of ACM SIGCOMM'95*, pages 288–298, August 1995.

[9] Digital Island. http://www.digitalisland.com.

[10] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. On the Placement of Internet Instrumentation. In *Proc. IEEE INFOCOM 2000*, March 2000.

[11] K.L. Johnson, J.F. Carr, M.S. Day, and M.F. Kaashoek. The Measured Performance of Content Distribution Networks. In *5th International Web Caching and Content Delivery Workshop*, May 2000.

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

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

[14] A. Myers and H.Zhang. Performance Characteristics of Mirror Servers on the Internet. In *Proceedings of IEEE INFOCOM'99*, March 1999.

[15] Napster. available from http://www.napster.com/.

[16] N. Natarajan, K.M. Hanna, and B.N. Levine. A Performance Study of Top Mirror Servers. Submitted for publication, January 2001.

[17] K. Obraczka and F. Silva. Network Latency Metrics for Server Proximity. In *Proc. of the IEEE Globecom 2000*, December 2000.

[18] C. Partridge, T. Mendez, and W. Milliken. Host Anycasting Service. IETF RFC 1546, November 1993.

[19] P. Rodriguez, A. Kirpal, and E. W. Biersack. Parallel-Access for Mirror Sites in the Internet. In *Proceedings of IEEE INFOCOM'00*, March 2000.

[20] M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek. Selection Algorithms for Replicated Web Servers. In *WorkShop on Internet Server Performance*, June 1998.

[21] Y. Shavitt, X. Sun, A. Wool, and B. Yener. Computing the Unmeasured: An Algebraic Approach to Internet Mapping. In *IEEE INFOCOM 2001*, April 2001.

Appendix A: Correlation Tables

	Hop count	10115	31800	107023	236694	524934	765736	1007102	Avg ping
Hop count	-	0.07	0.07	0.08	0.09	0.10	0.11	0.11	0.07
10115	0.07	-	0.35	0.33	0.32	0.32	0.32	0.31	0.16
31800	0.07	0.35	-	0.61	0.58	0.57	0.55	0.53	0.20
107023	0.08	0.33	0.61	-	0.79	0.77	0.74	0.72	0.21
236694	0.09	0.32	0.58	0.79	-	0.87	0.82	0.80	0.22
524934	0.10	0.32	0.57	0.77	0.87	-	0.90	0.88	0.21
765736	0.11	0.32	0.55	0.74	0.82	0.90	-	0.92	0.21
1007102	0.11	0.31	0.53	0.72	0.80	0.88	0.92	-	0.20
Avg ping	0.07	0.16	0.20	0.21	0.22	0.21	0.21	0.20	-

Figure 20: Correlation table for UCSC.

	Hop count	10115	31800	107023	236694	524934	765736	1007102	Avg ping
Hop count	-	0.06	0.06	0.06	0.04	0.04	0.06	0.05	0.19
10115	0.06	-	0.27	0.31	0.29	0.28	0.31	0.29	0.18
31800	0.06	0.27	-	0.37	0.41	0.34	0.37	0.36	0.23
107023	0.06	0.31	0.37	-	0.50	0.51	0.53	0.50	0.23
236694	0.04	0.29	0.41	0.50	-	0.60	0.55	0.60	0.23
524934	0.04	0.28	0.34	0.51	0.60	-	0.72	0.69	0.21
765736	0.06	0.31	0.37	0.53	0.55	0.72	-	0.72	0.22
1007102	0.05	0.29	0.36	0.50	0.60	0.69	0.72	-	0.22
Avg ping	0.19	0.18	0.23	0.23	0.23	0.21	0.22	0.22	-

Figure 21: Correlation table for Purdue.

	Hop count	10115	31800	107023	236694	524934	765736	1007102	Avg ping
Hop count	-	0.08	0.07	0.09	0.11	0.12	0.12	0.12	0.11
10115	0.08	-	0.33	0.36	0.37	0.37	0.36	0.36	0.14
31800	0.07	0.33	-	0.58	0.56	0.56	0.55	0.54	0.21
107023	0.09	0.36	0.58	-	0.78	0.78	0.75	0.74	0.26
236694	0.11	0.37	0.56	0.78	-	0.88	0.85	0.84	0.30
524934	0.12	0.37	0.56	0.78	0.88	-	0.92	0.89	0.30
765736	0.12	0.36	0.55	0.75	0.85	0.92	-	0.93	0.30
1007102	0.12	0.36	0.54	0.74	0.84	0.89	0.93	-	0.29
Avg ping	0.11	0.14	0.21	0.26	0.30	0.30	0.30	0.29	-

Figure 22: Correlation table for UDel.

	Hop count	10115	31800	107023	236694	524934	765736	1007102	Avg ping
Hop count	-	0.05	0.06	0.07	0.06	0.07	0.07	0.08	0.09
10115	0.05	-	0.17	0.21	0.22	0.25	0.24	0.24	0.21
31800	0.06	0.17	-	0.33	0.33	0.33	0.35	0.34	0.24
107023	0.07	0.21	0.33	-	0.53	0.54	0.55	0.55	0.34
236694	0.06	0.22	0.33	0.53	-	0.67	0.67	0.66	0.39
524934	0.07	0.25	0.33	0.54	0.67	-	0.80	0.79	0.45
765736	0.07	0.24	0.35	0.55	0.67	0.80	-	0.85	0.46
1007102	0.08	0.24	0.34	0.55	0.66	0.79	0.85	-	0.45
Avg ping	0.09	0.21	0.24	0.34	0.39	0.45	0.46	0.45	-

Figure 23: Correlation table for UMass.

	Hop count	10115	31800	107023	236694	524934	765736	1007102	Avg ping
Hop count	-	0.18	0.16	0.15	0.15	0.14	0.14	0.14	0.28
10115	0.18	-	0.44	0.46	0.45	0.45	0.45	0.44	0.39
31800	0.16	0.44	-	0.70	0.69	0.68	0.67	0.66	0.49
107023	0.15	0.46	0.70	-	0.83	0.83	0.81	0.80	0.49
236694	0.15	0.45	0.69	0.83	-	0.90	0.88	0.87	0.50
524934	0.14	0.45	0.68	0.83	0.90	-	0.93	0.91	0.48
765736	0.14	0.45	0.67	0.81	0.88	0.93	-	0.95	0.47
1007102	0.14	0.44	0.66	0.80	0.87	0.91	0.95	-	0.45
Avg ping	0.28	0.39	0.49	0.49	0.50	0.48	0.47	0.45	-

Figure 24: Correlation table for UNC.

	Hop count	10115	31800	107023	236694	524934	765736	1007102	Avg ping
Hop count	-	0.07	0.10	0.09	0.09	0.09	0.09	0.10	0.23
10115	0.07	-	0.30	0.34	0.36	0.35	0.36	0.35	0.25
31800	0.10	0.30	-	0.57	0.55	0.55	0.55	0.55	0.34
107023	0.09	0.34	0.57	-	0.76	0.76	0.76	0.75	0.38
236694	0.09	0.36	0.55	0.76	-	0.86	0.85	0.84	0.40
524934	0.09	0.35	0.55	0.76	0.86	-	0.91	0.90	0.40
765736	0.09	0.36	0.55	0.76	0.85	0.91	-	0.92	0.41
1007102	0.10	0.35	0.55	0.75	0.84	0.90	0.92	-	0.40
Avg ping	0.23	0.25	0.34	0.38	0.40	0.40	0.41	0.40	-

Figure 25: Correlation table for USC.

Appendix B: More Results

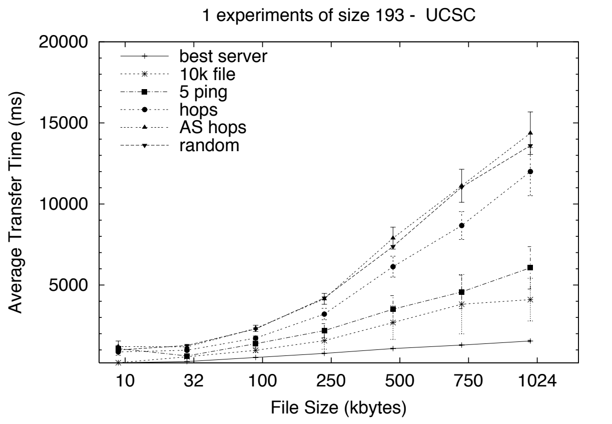

Figure 26: Performance of server selection metrics (UCSC, 193-set)

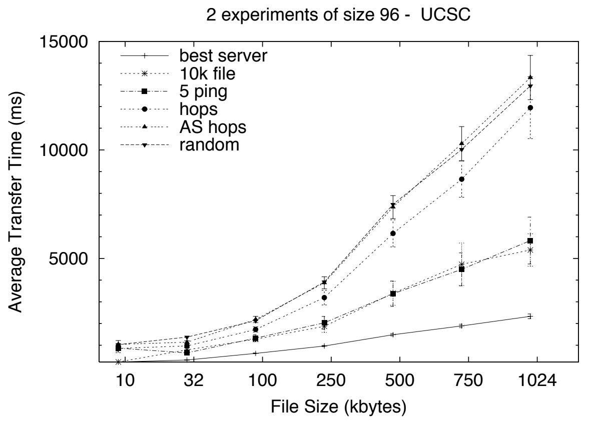

Figure 27: Performance of server selection metrics (UCSC, 96-subset)

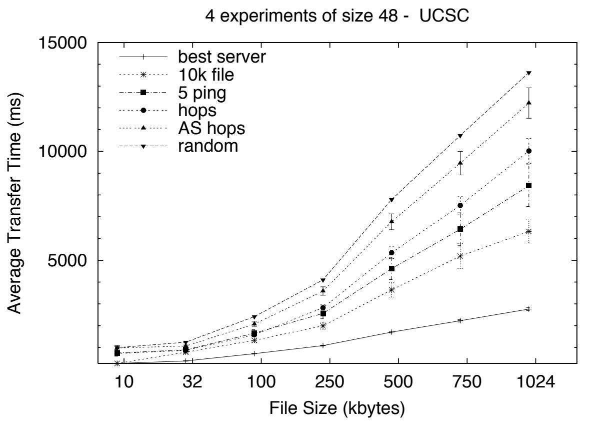

Figure 28: Performance of server selection metrics (UCSC, 48-subset)

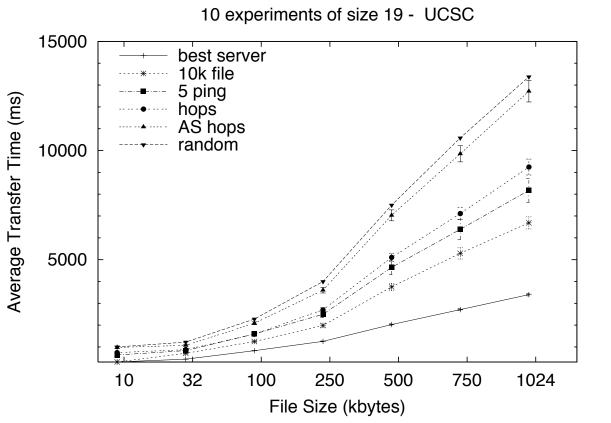

Figure 29: Performance of server selection metrics (UCSC, 19-subset)

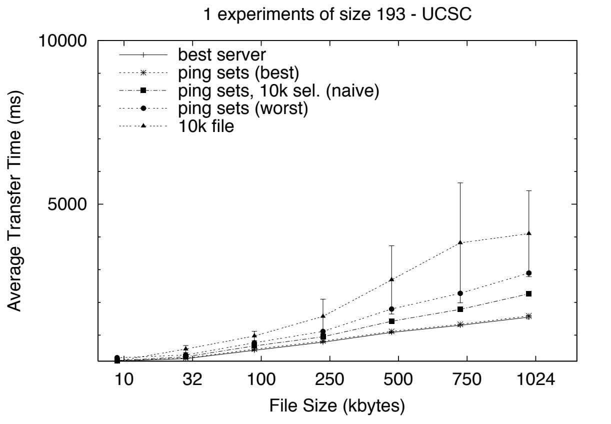

Figure 30: Performance of ping sets $n = 5$ (UCSC, 193-set).

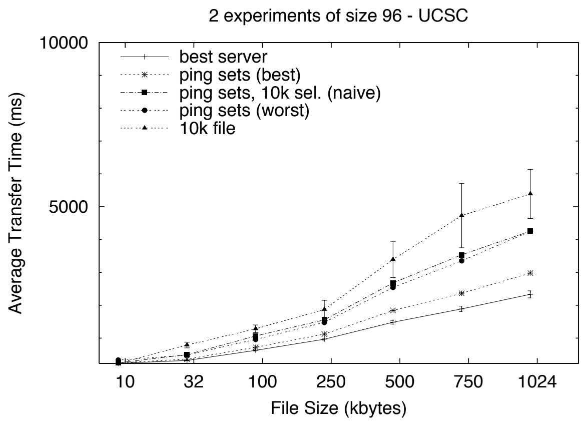

Figure 31: Performance of ping sets $n = 5$ (UCSC, 96-subset).

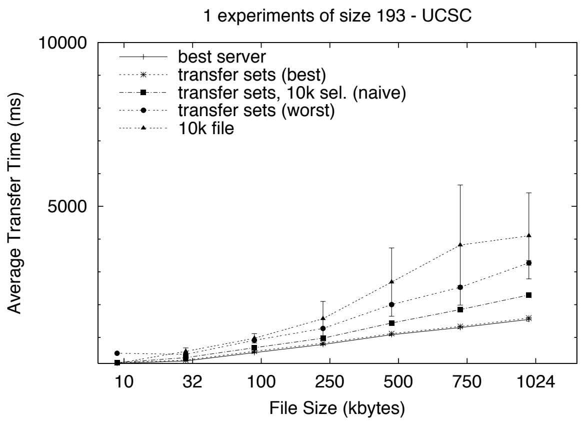

Figure 34: Performance of transfer sets $n = 5, m = 3$ (UCSC, 193-set).

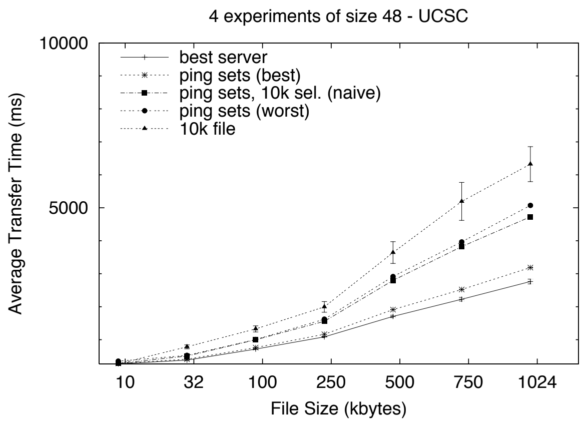

Figure 32: Performance of ping sets $n = 5$ (UCSC, 48-subset).

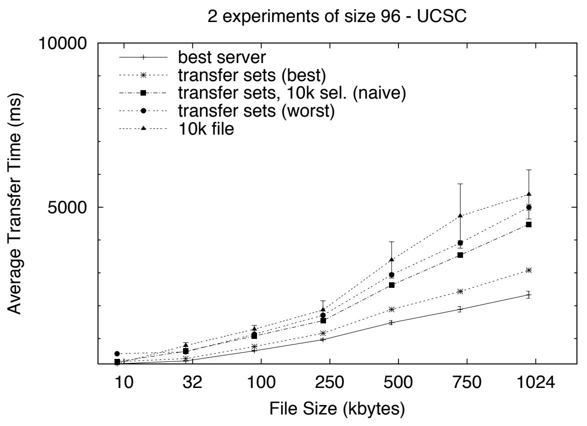

Figure 35: Performance of transfer sets $n = 5, m = 3$ (UCSC, 96-subset).

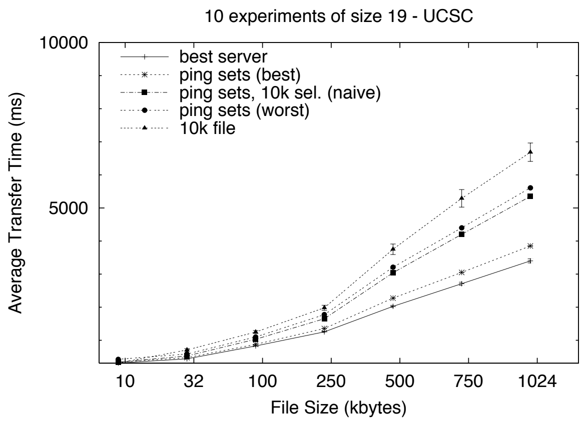

Figure 33: Performance of ping sets $n = 5$ (UCSC, 19-subset).

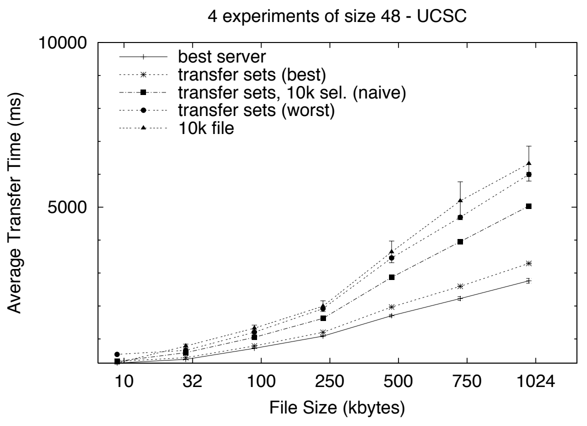

Figure 36: Performance of transfer sets $n = 5, m = 3$ (UCSC, 48-subset).

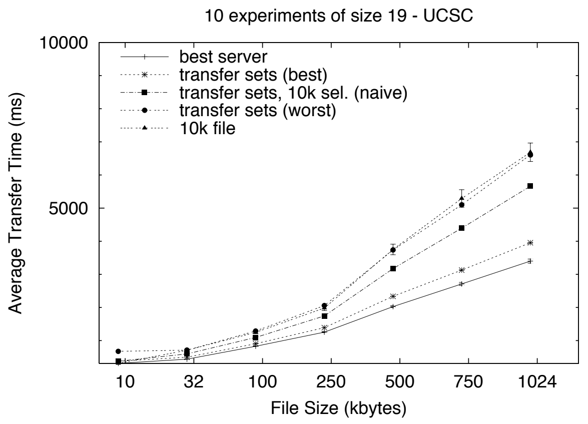

Figure 37: Performance of transfer sets $n = 5, m = 3$ (UCSC, 19-subset).

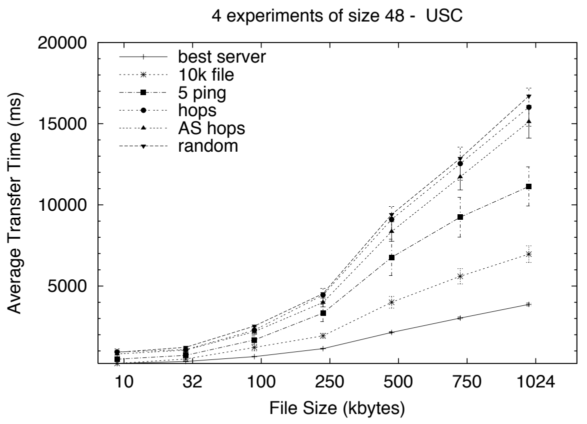

Figure 40: Performance of server selection metrics (USC, 48-subset)

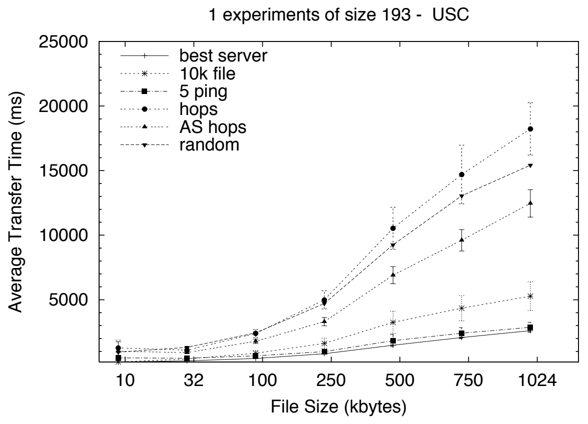

Figure 38: Performance of server selection metrics (USC, 193-set)

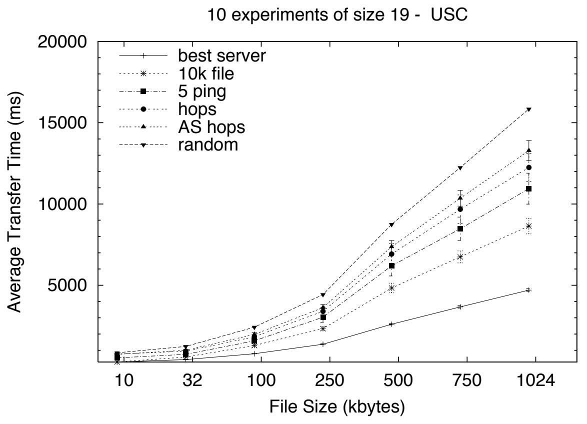

Figure 41: Performance of server selection metrics (USC, 19-subset)

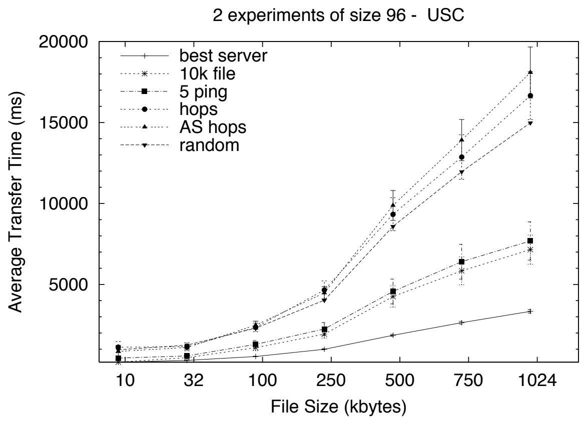

Figure 39: Performance of server selection metrics (USC, 96-subset)

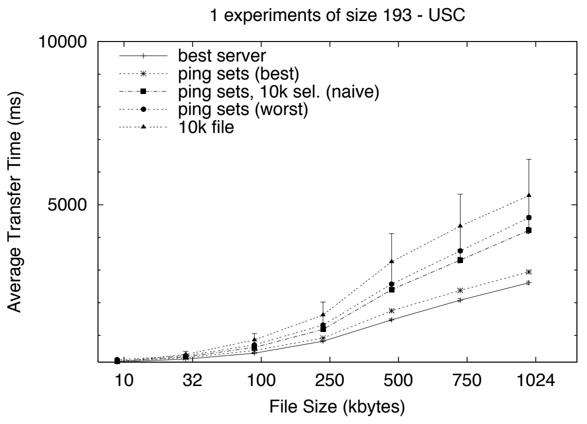

Figure 42: Performance of ping sets $n = 5$ (USC, 193-set).

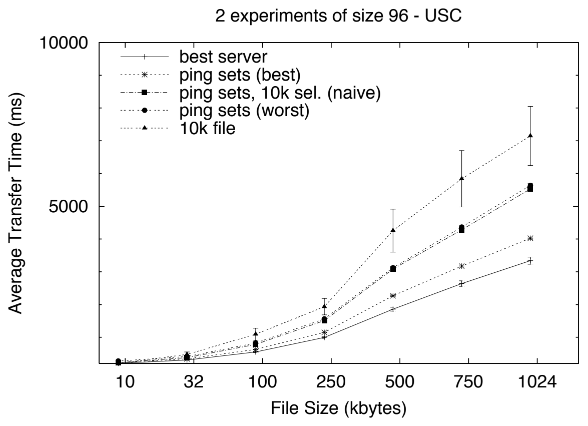

Figure 43: Performance of ping sets $n = 5$ (USC, 96-subset).

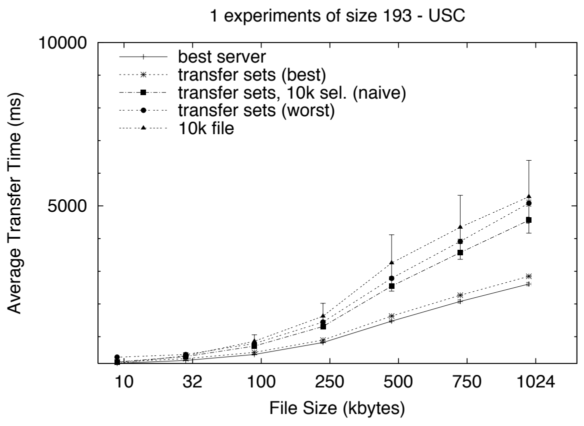

Figure 46: Performance of transfer sets $n = 5, m = 3$ (USC, 193-set).

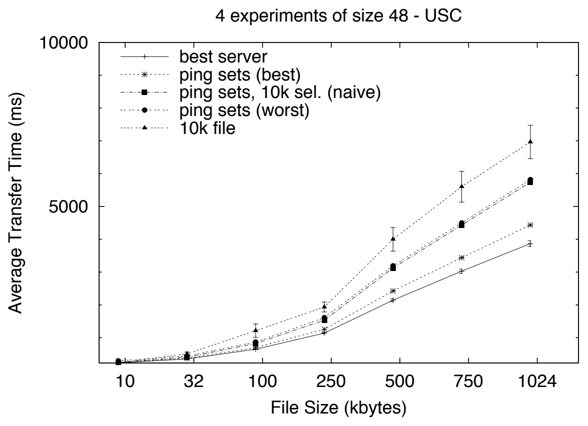

Figure 44: Performance of ping sets $n = 5$ (USC, 48-subset).

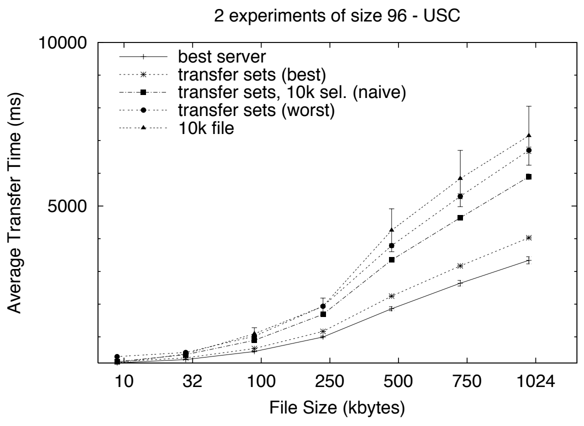

Figure 47: Performance of transfer sets $n = 5, m = 3$ (USC, 96-subset).

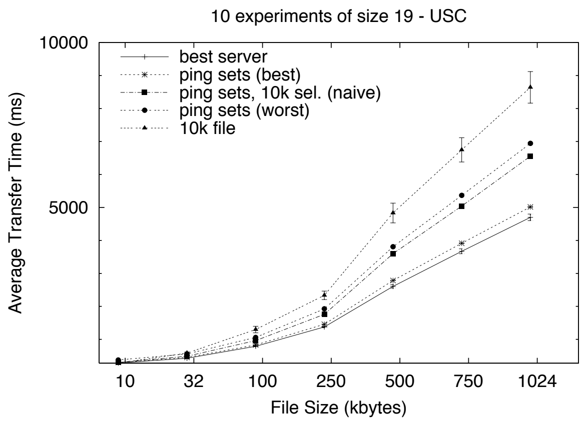

Figure 45: Performance of ping sets $n = 5$ (USC, 19-subset).

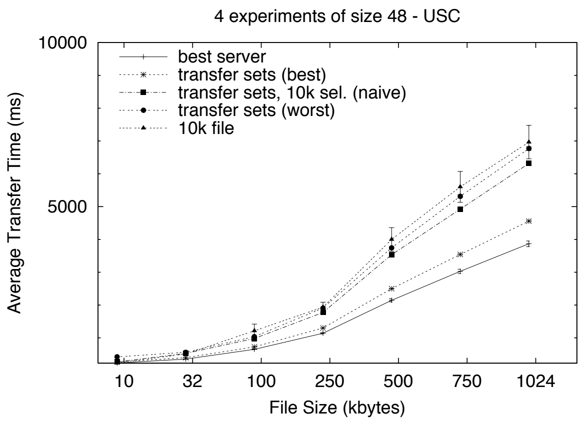

Figure 48: Performance of transfer sets $n = 5, m = 3$ (USC, 48-subset).

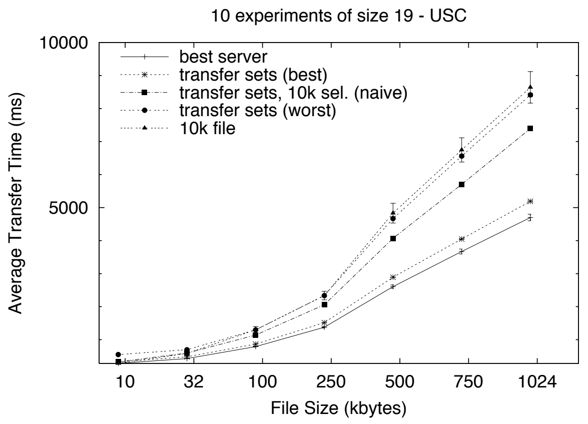

Figure 49: Performance of transfer sets $n = 5, m = 3$ (USC, 19-subset).

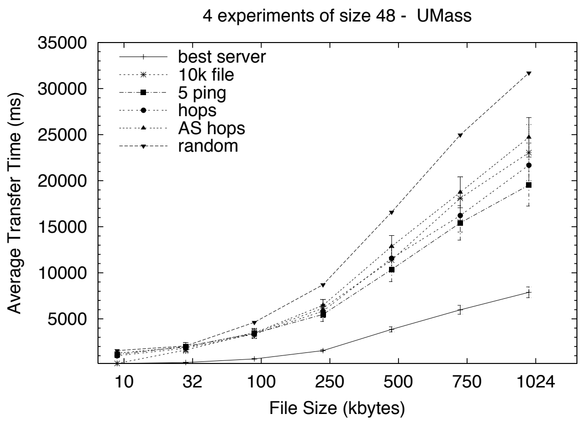

Figure 52: Performance of server selection metrics (UMass, 48-subset)

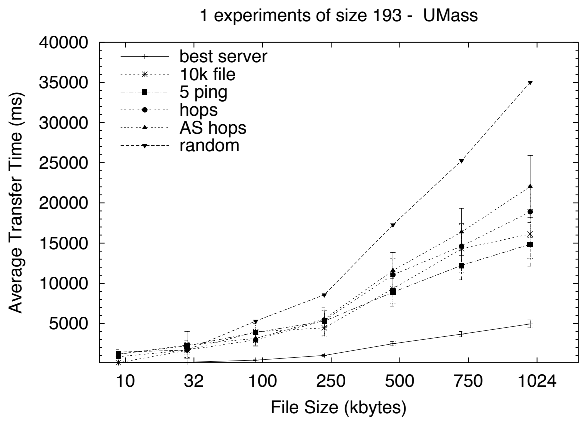

Figure 50: Performance of server selection metrics (UMass, 193-set)

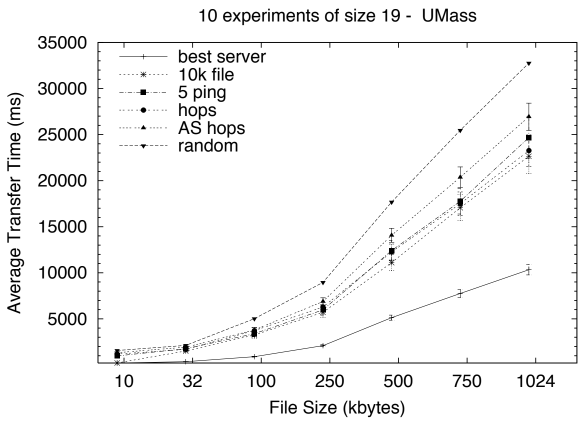

Figure 53: Performance of server selection metrics (UMass, 19-subset)

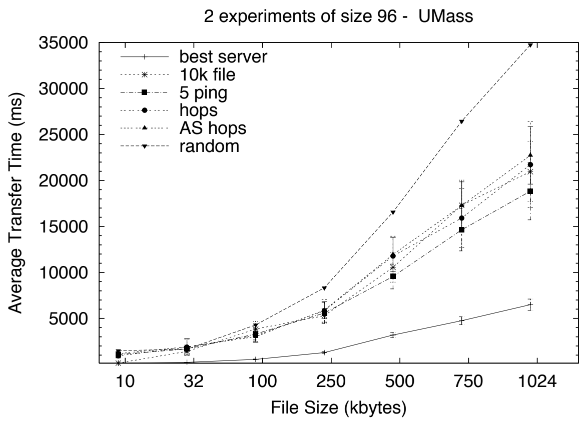

Figure 51: Performance of server selection metrics (UMass, 96-subset)

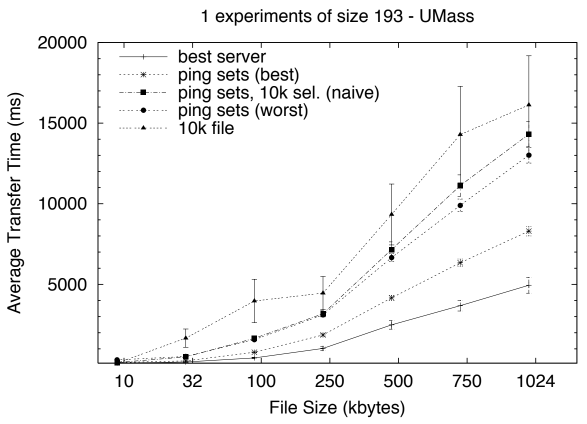

Figure 54: Performance of ping sets $n = 5$ (UMass, 193-set).

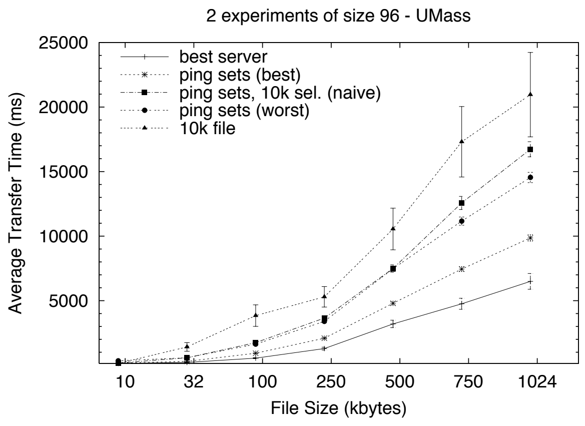

Figure 55: Performance of ping sets $n = 5$ (UMass, 96-subset).

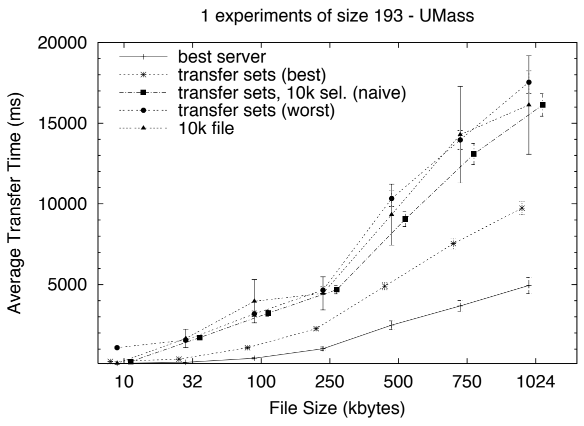

Figure 58: Performance of transfer sets $n = 5, m = 3$ (UMass, 193-set).

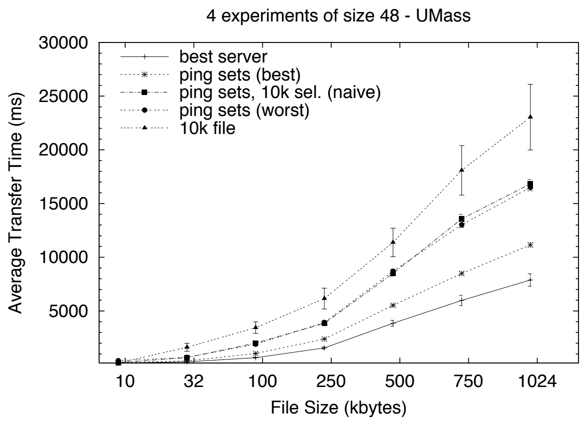

Figure 56: Performance of ping sets $n = 5$ (UMass, 48-subset).

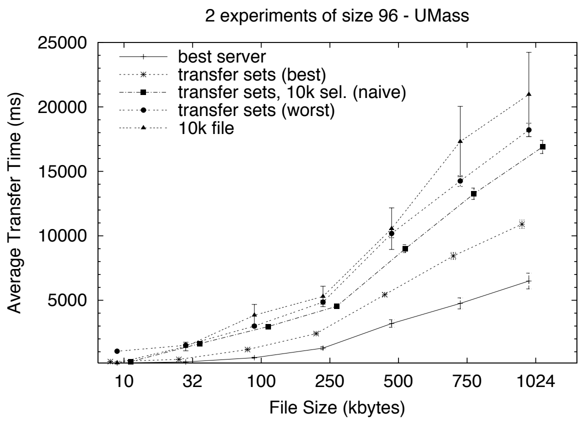

Figure 59: Performance of transfer sets $n = 5, m = 3$ (UMass, 96-subset).

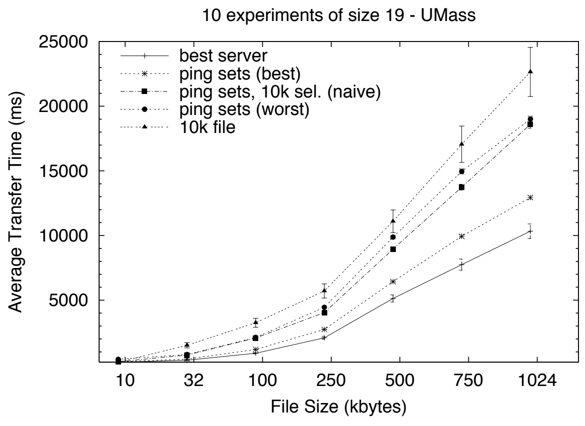

Figure 57: Performance of ping sets $n = 5$ (UMass, 19-subset).

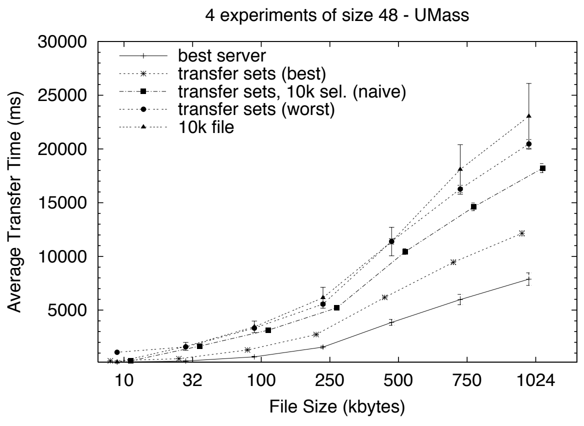

Figure 60: Performance of transfer sets $n = 5, m = 3$ (UMass, 48-subset).

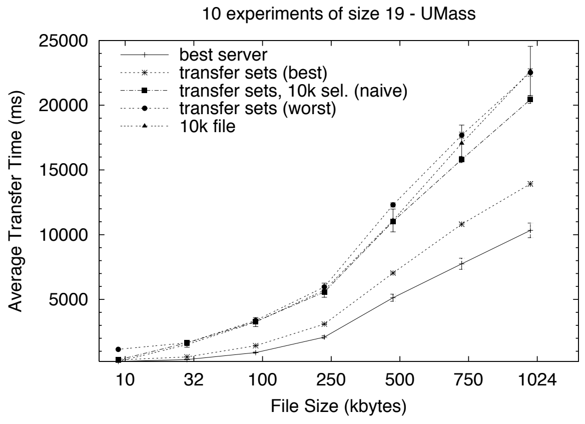

Figure 61: Performance of transfer sets $n = 5, m = 3$ (UMass, 19-subset).

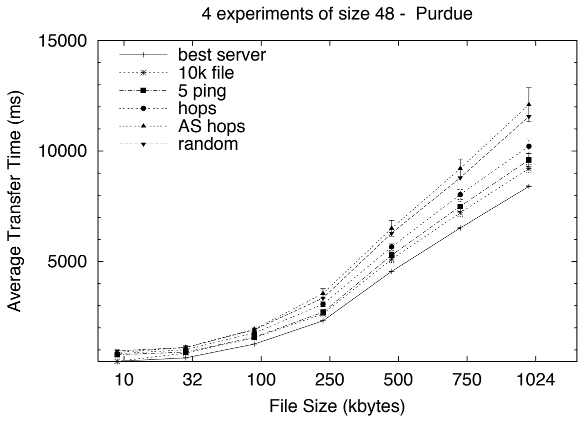

Figure 64: Performance of server selection metrics (Purdue, 48-subset)

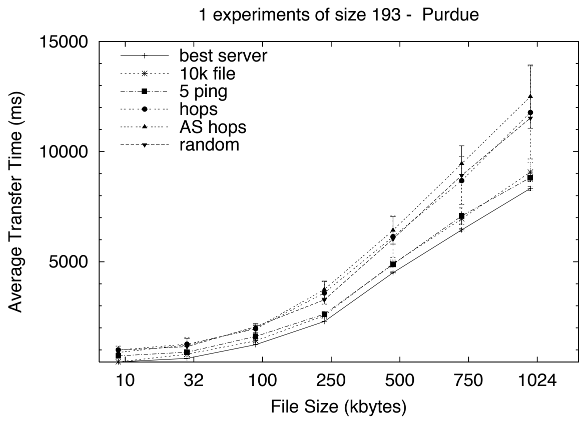

Figure 62: Performance of server selection metrics (Purdue, 193-set)

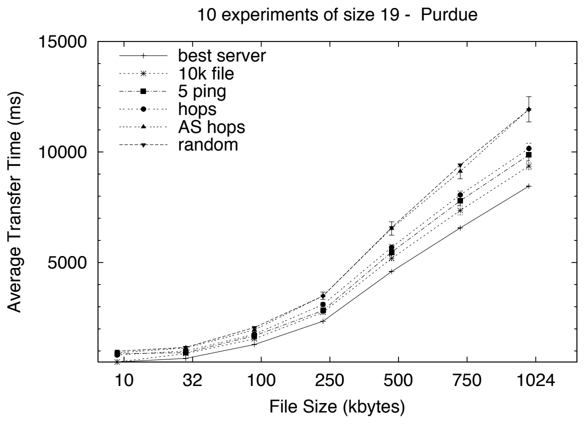

Figure 65: Performance of server selection metrics (Purdue, 19-subset)

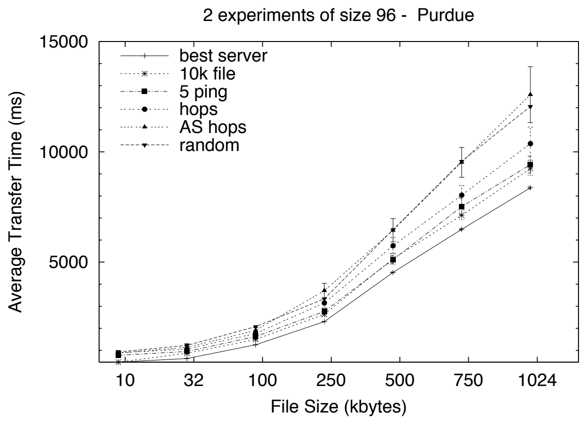

Figure 63: Performance of server selection metrics (Purdue, 96-subset)

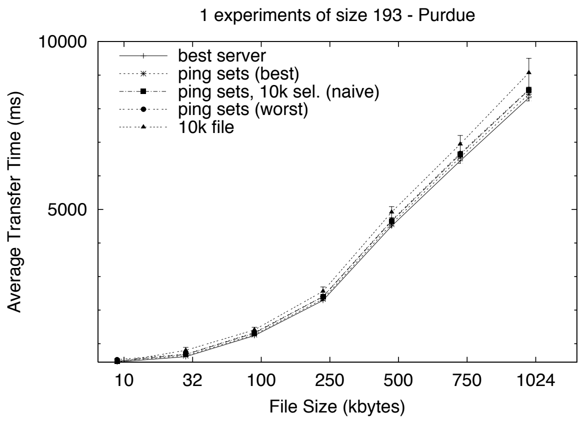

Figure 66: Performance of ping sets $n = 5$ (Purdue, 193-set).

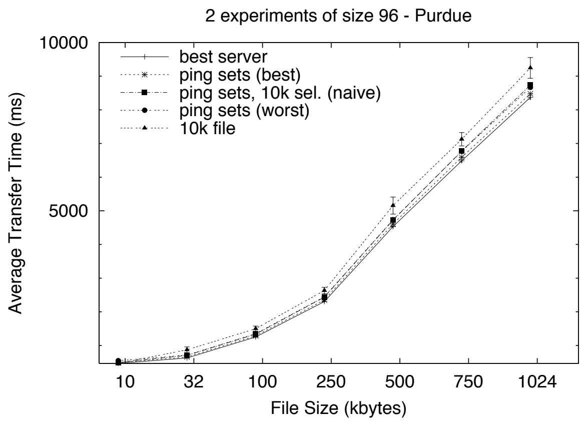

Figure 67: Performance of ping sets $n = 5$ (Purdue, 96-subset).

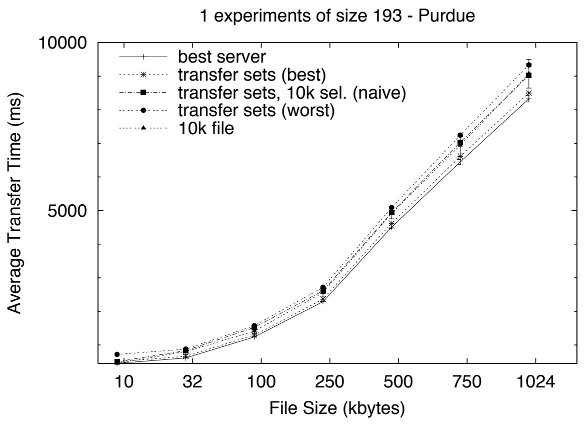

Figure 70: Performance of transfer sets $n = 5, m = 3$ (Purdue, 193-set).

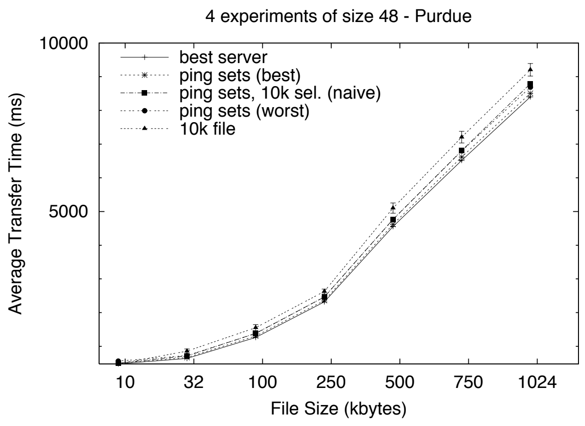

Figure 68: Performance of ping sets $n = 5$ (Purdue, 48-subset).

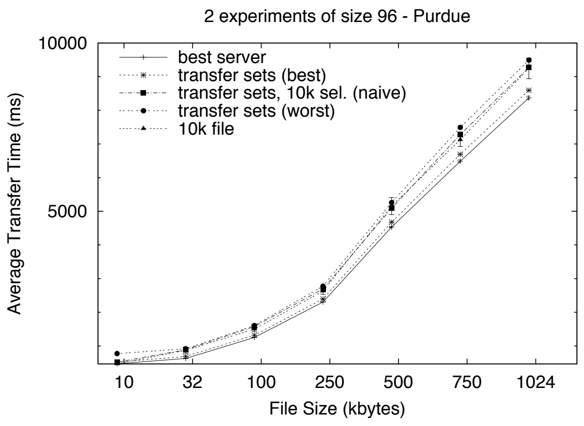

Figure 71: Performance of transfer sets $n = 5, m = 3$ (Purdue, 96-subset).

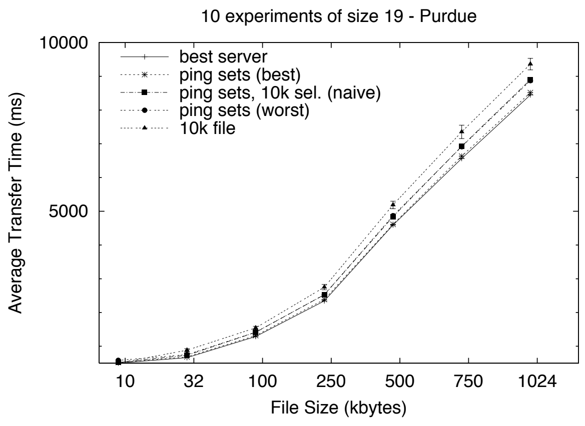

Figure 69: Performance of ping sets $n = 5$ (Purdue, 19-subset).

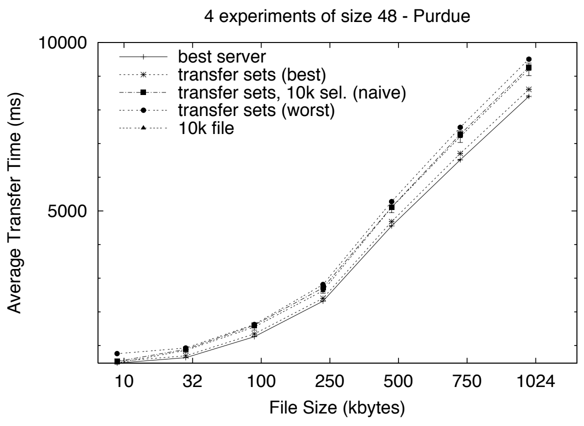

Figure 72: Performance of transfer sets $n = 5, m = 3$ (Purdue, 48-subset).

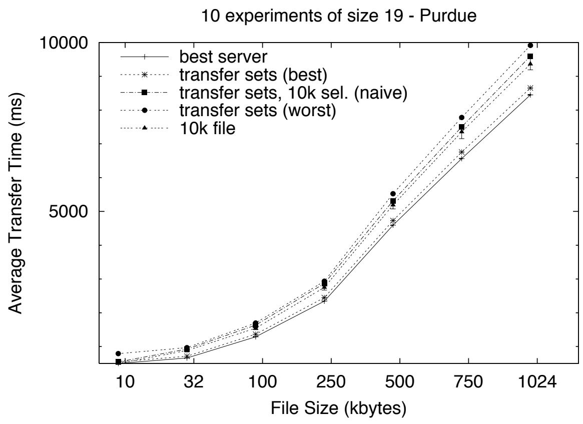

Figure 73: Performance of transfer sets $n = 5, m = 3$ (Purdue, 19-subset).

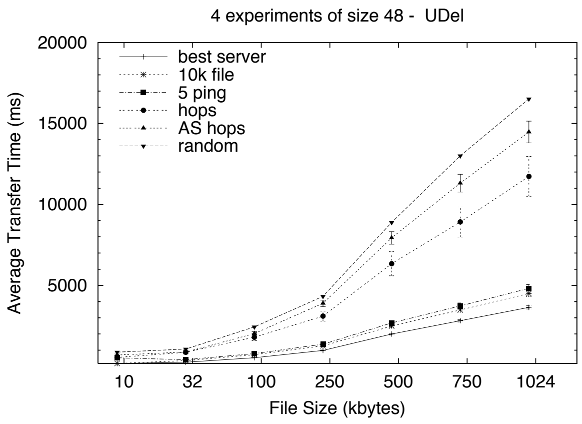

Figure 76: Performance of server selection metrics (UDel, 48-subset)

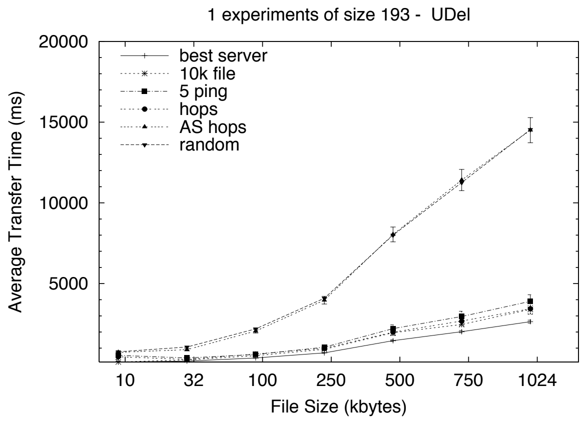

Figure 74: Performance of server selection metrics (UDel, 193-set)

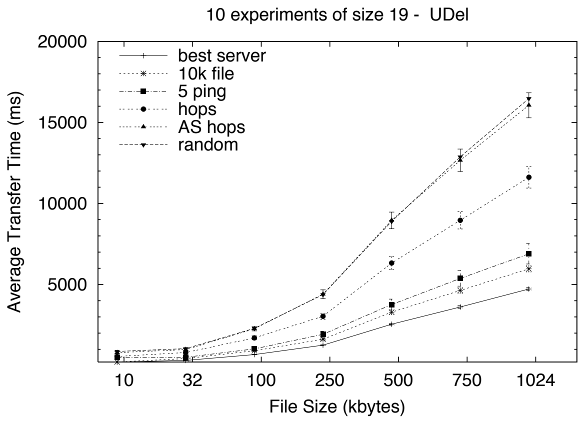

Figure 77: Performance of server selection metrics (UDel, 19-subset)

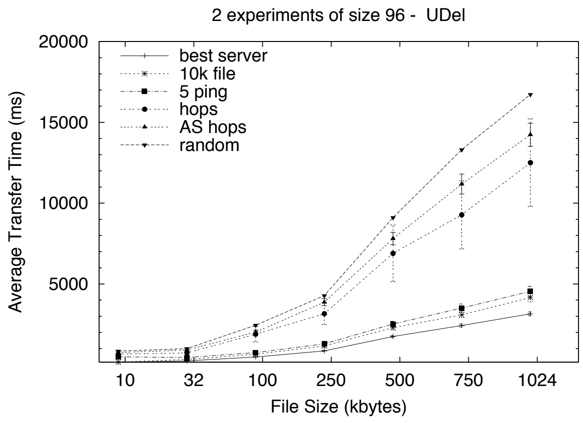

Figure 75: Performance of server selection metrics (UDel, 96-subset)

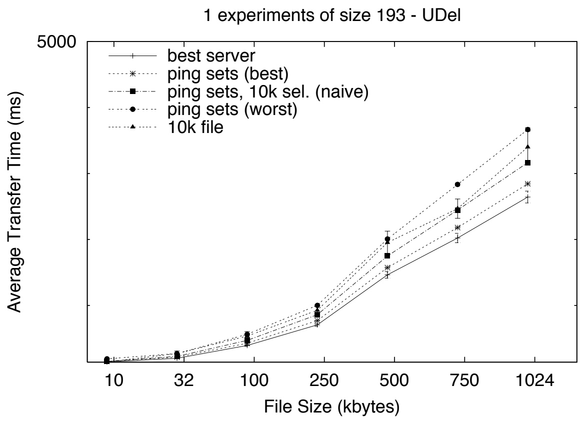

Figure 78: Performance of ping sets $n = 5$ (UDel, 193-set).

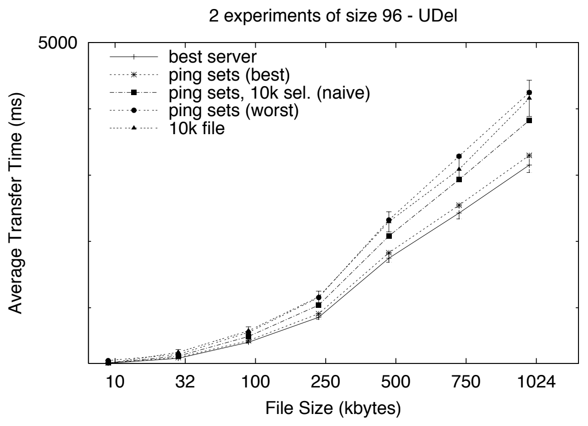

Figure 79: Performance of ping sets $n = 5$ (UDel, 96-subset).

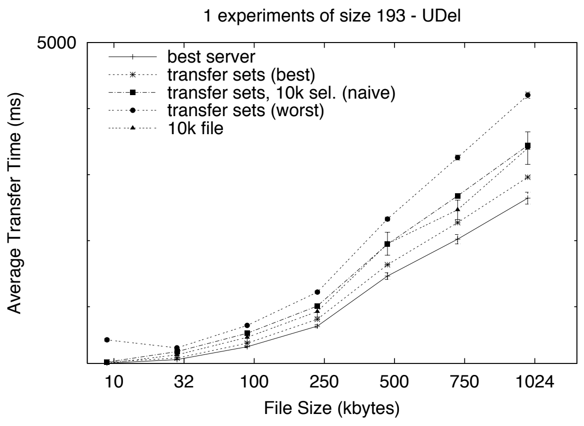

Figure 82: Performance of transfer sets $n = 5, m = 3$ (UDel, 193-set).

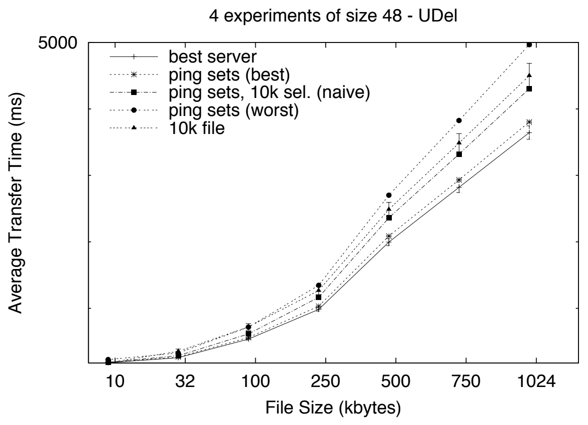

Figure 80: Performance of ping sets $n = 5$ (UDel, 48-subset).

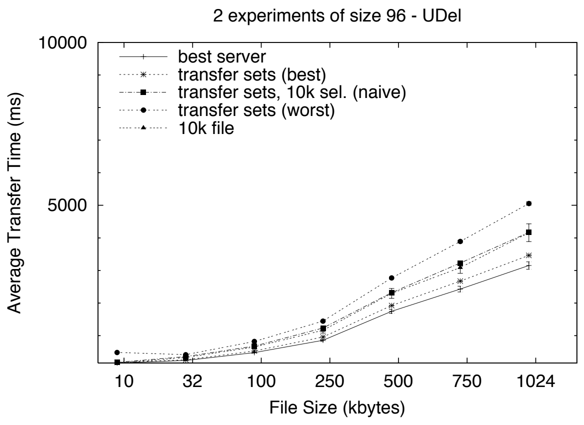

Figure 83: Performance of transfer sets $n = 5, m = 3$ (UDel, 96-subset).

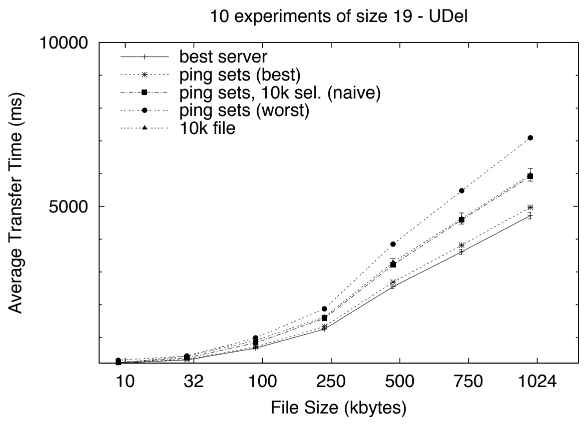

Figure 81: Performance of ping sets $n = 5$ (UDel, 19-subset).

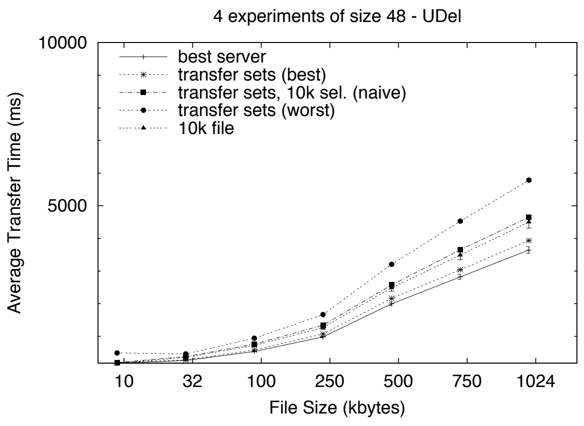

Figure 84: Performance of transfer sets $n = 5, m = 3$ (UDel, 48-subset).

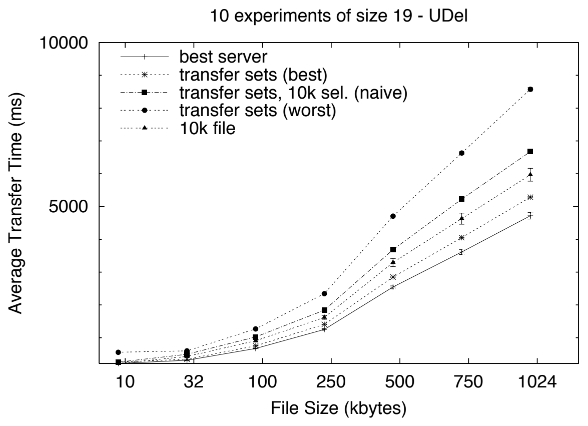

Figure 85: Performance of transfer sets $n = 5, m = 3$ (UDel, 19-subset).

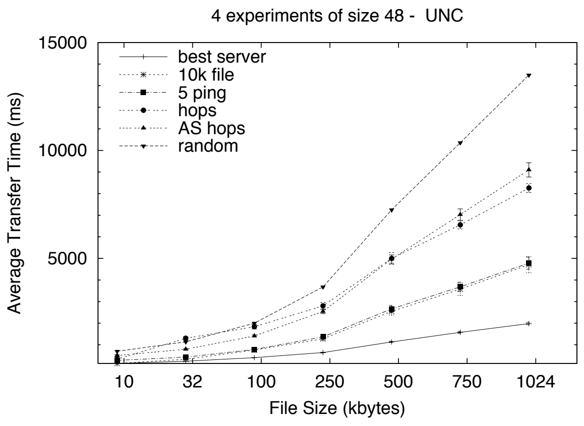

Figure 88: Performance of server selection metrics (UNC, 48-subset)

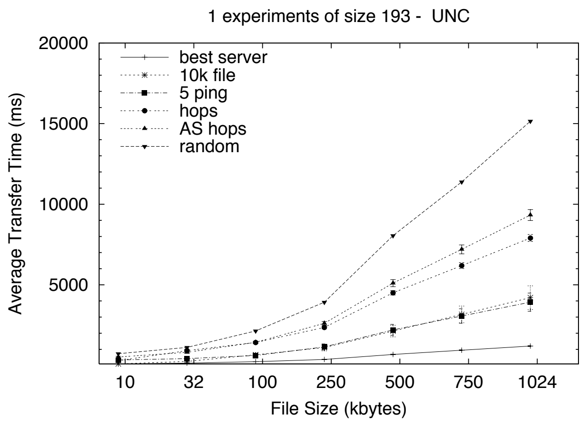

Figure 86: Performance of server selection metrics (UNC, 193-set)

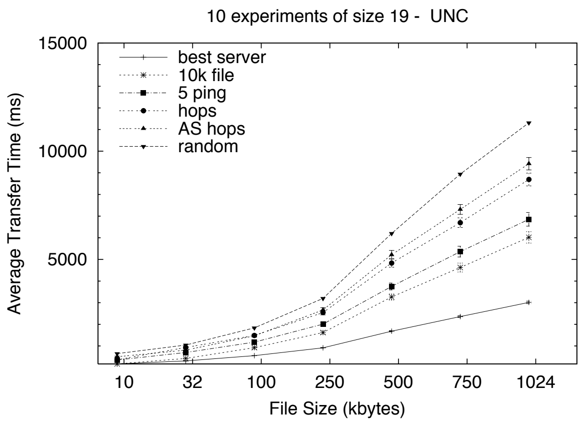

Figure 89: Performance of server selection metrics (UNC, 19-subset)

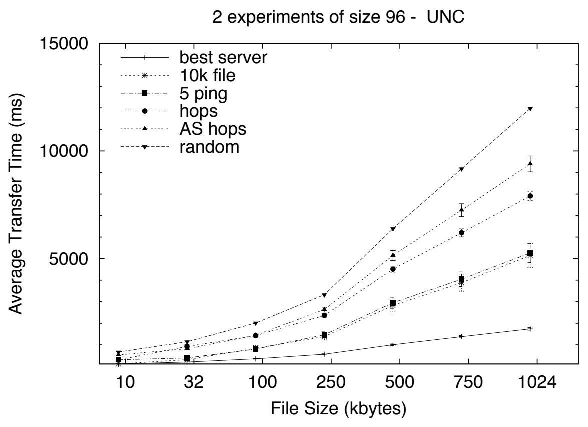

Figure 87: Performance of server selection metrics (UNC, 96-subset)

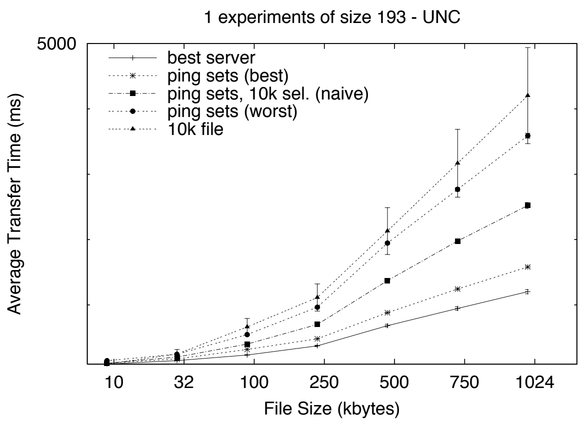

Figure 90: Performance of ping sets $n = 5$ (UNC, 193-set).

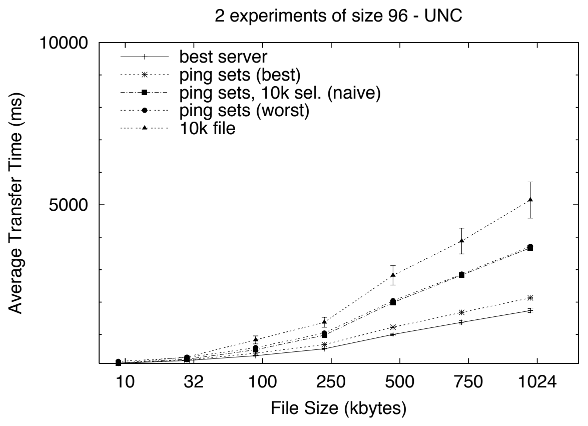

Figure 91: Performance of ping sets $n = 5$ (UNC, 96-subset).

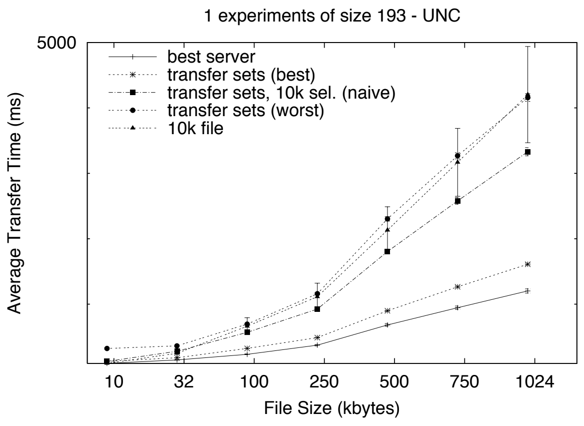

Figure 94: Performance of transfer sets $n = 5, m = 3$ (UNC, 193-set).

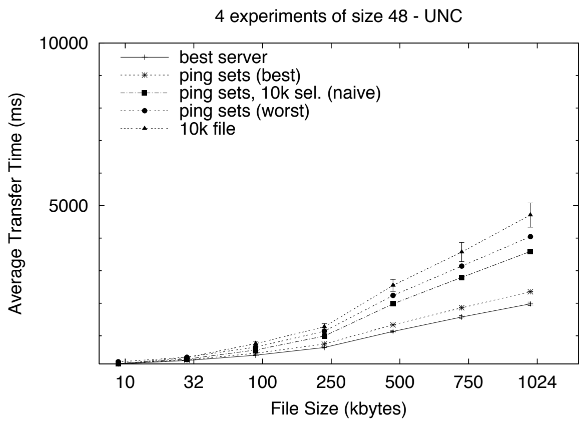

Figure 92: Performance of ping sets $n = 5$ (UNC, 48-subset).

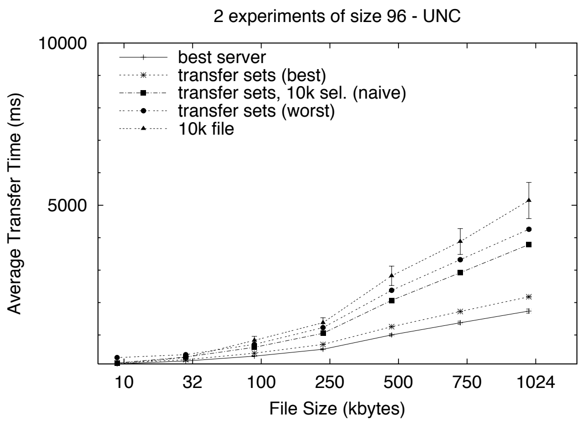

Figure 95: Performance of transfer sets $n = 5, m = 3$ (UNC, 96-subset).

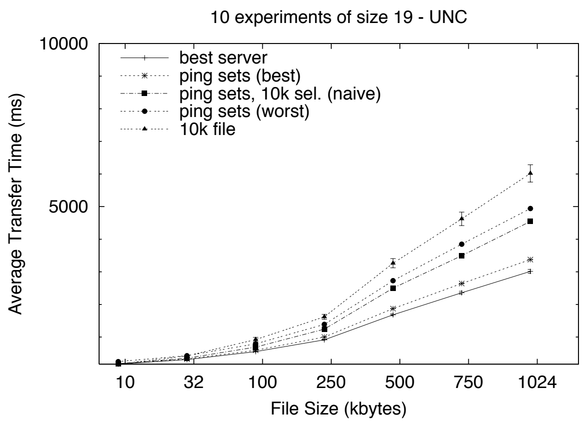

Figure 93: Performance of ping sets $n = 5$ (UNC, 19-subset).

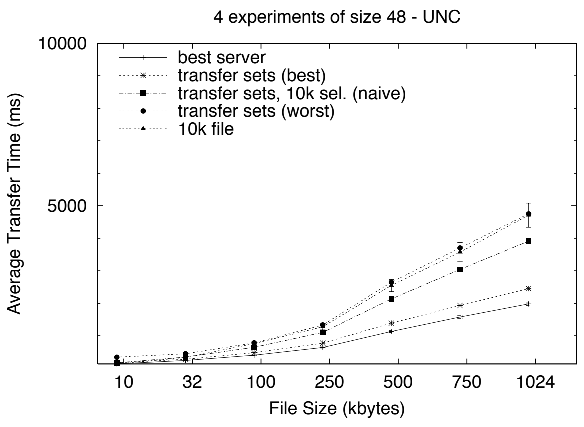

Figure 96: Performance of transfer sets $n = 5, m = 3$ (UNC, 48-subset).

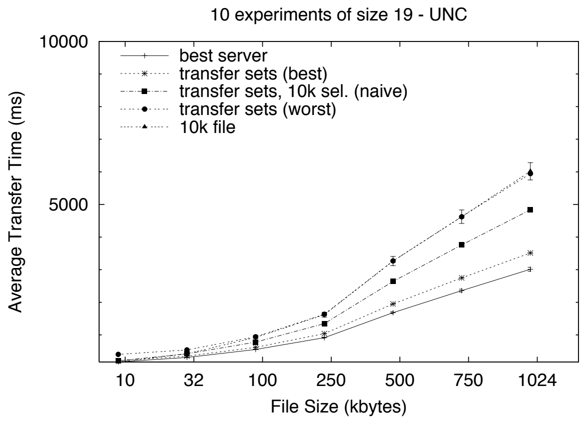

Figure 97: Performance of transfer sets $n = 5, m = 3$ (UNC, 19-subset).