6.894 Final Project
Ben Fu, Mike Feng, Victor Chang

Performance Evaluation of a Replica Selection Scheme That Incorporates Bandwidth and Latency

Abstract

The explosion in the number of World Wide Web subscribers has forced many online service providers to distribute their systems into multiple replicas spread throughout the Internet to handle the large volume of requests. Efficient replica selection schemes reduce the server response time experienced by the users. Many selection schemes exist today, but few incorporate real-time bandwidth information in their selection because of the time overhead involved in bandwidth measurement. This paper introduces *two-phase probe*, a selection scheme that utilizes both latency and bandwidth information in selecting a replica. We test the performance of this scheme by comparing it with three benchmark schemes: *dynamic probe, dynamic bandwidth,* and *round robin*. The four schemes are simultaneously evaluated in an environment of five web replicas distributed across the country. Our results show that dynamic probe performed the best of all schemes. We believe that two-phase probe performs worse than dynamic probe because of inaccurate bandwidth measurements; and if better bandwidth measurement methods are used, the scheme may perform better.

1. Introduction

Global usage of the WWW has increased drastically over the past five years, and consequently, many web servers today encounter heavy traffic. To attain high performance and high availability in this environment, larger web sites have employed the concept of content replication over replica servers. These web sites deploy multiple replica servers into geographically diverse regions and place the same content on each replica. Replicas provide the benefit of load distribution, fault tolerance, and client proximity. A client accessing such a web site would be transparently directed to one server amongst the replicas through a selection scheme. In order to realize the high performance and availability made possible by replication, the server selection scheme must intelligently select the server that could complete the client's requests the soonest. The key question behind how to select the best server is: what metric should be used to compare the replicas? Once a metric has been decided upon, the client only has to choose the replica that fares best in the comparison. Of course, the best metric is the amount of time required for the server to carry out the client's request. However, due to varying network and server conditions, that metric cannot be measured until a request has been completed! One can only attempt to predict that response time using other observable characteristics.

Much research has been performed in developing server selection schemes because a good scheme produces huge benefits by providing low response times to clients. A server's response time can be broken down into the *connection time* to the server, the *latency time* for the server to respond, and the *transmission time* for the server to complete the transmission of all the data to the client [7]. Most of the server schemes use some version of the following metrics, and different metrics reveal different portions of server response time:

- The *hop count* from client to server. One of the main reasons that replica servers are often geographically distributed is the notion that clients can receive better service from servers located nearest to them. The hop count between the client and a replica should indicate the proximity between them, and thereby indirectly indicate the connection time and latency time to the server.
- The *probe latency*. The client obtains this metric by sending out an echo datagram to each replica, and the roundtrip response latency of each datagram is used as the metric. This metric reveals the connection time to the server plus the latency time for one packet to travel.
- The *path bandwidth*. One probe datagram may not reveal the bandwidth available in the path between the client and a replica. In the transfer of large files, the path bandwidth is an important factor in determining the server's response time. It indicates the data transmission time in the path.
- The *Server load*. Besides the network, the replica servers themselves contribute to the response time to a request. Its load could affect all parts of the response time.

How do these metrics compare? Hop count and probe latency both approximate the length of the path between the client and server, but probe latency encapsulates information about real-time network conditions while hop count does not. Past research has shown that probe latency is a better metric than hop count [4]. Nonetheless, because the transfer of one probe datagram is very insignificant relative to the capacity of most networks, probe latency is not sufficient to assess the transmission time of files of significant size between the client and the server. In the WWW environment where large file transfers occur frequently, information about the path bandwidth should be taken into account by selection schemes. Unfortunately, bandwidth measurement requires much more time than probe latency measurement, and it introduces extra time cost into the selection process. A server's load could be a factor in determining its response time; however, server load is not necessarily a useful metric in predicting response time. We assume that different replicas of a web site could have varying levels of capabilities; in that case, a server under a heavy load may still offer higher performance than and be preferred over a less burdened but less capable server. Probe latency and path bandwidth encapsulate server performance in them, since the server is itself a link in the network path. A metric that specifically addresses server load cannot offer much more information about its potential performance. The only situation in which a server's load becomes a significant factor in replica selection is when the server has reached its maximum capacity and begins to ignore requests. When that happens, it should be avoided. Thus, server load can be used as a filtering threshold. If a replica's load exceeds its threshold, it should be eluded; if not, it's load should be ignored.

In this paper, we propose *two-phase probe* – a client-side server selection scheme that utilizes both probe latency and path bandwidth in its metric. The scheme combines the two schemes by making two assumptions of WWW user behaviors. First, most WWW users consecutively make more than one page access during a visit to a web site. Users usually first access a "main" page and then browse through a few links that get them other files from the same web site [1]. Second, we assume that larger file transfers from a web site does not normally occur in the first page a user reads from a web site, which is usually a main page with small content. This means that the accuracy of a replica selection scheme is not as important to the user's first page access as it is to his subsequent accesses on the same web site. We take advantage of these assumptions and split our replica selection scheme into two phases. The first phase optimizes speed by using probe latency for selection, and the second phase optimizes accuracy by incorporating path bandwidth in the selection.

The remainder of this paper describes some of the existing replica selection schemes – much of our design derives from these schemes. The paper discusses how two-phase probe improves upon the existing schemes. Section 2 explains the system in detail – its metrics and its architecture. Section 3 describes the process with which we compare two-phase probe against other benchmark replica selection schemes. Lastly, Section 4 analyzes the results of our comparisons.

1.1 Design Goals

We designed two-phase probe with the following goals in mind:

- The scheme should consistently select the server that will provide the lowest response time to the client's request. This is the primary goal of replica selection that all selection schemes share.
- The selection scheme should minimize delays observable to the user. The user could see such delays as part of the server response time. In alignment to the first goal listed above, the perceived server response time should be minimized.
- The selection process must remain transparent to the user. It should appear to the user as if the client is always accessing a single server. The user should not realize that it is being directed to one of several replicas.
- The selection scheme should have minimal implementation requirements. We want the scheme to require as little hardware or infrastructure modifications as possible, so systems can easily implement the scheme and remain flexible.

2. Related Work

Existing replica selection schemes can be categorized into two ways. First, it can be categorized by the nature of its selection metrics: *static, statistical,* or *dynamic*. Static metrics assume that network conditions do not fluctuate. They're based on values that reflect the network or server configurations, such as hop count and base bandwidth. This category of metrics indicates the potential of the replicas and the network, but they reveal nothing about current traffic going through them. Statistical metrics assume that network conditions only fluctuate slightly, so they derive from a statistical combination of past measurements. For example, a statistical probe latency metric sends a probe upon every client request, but arrives at the metric by averaging its value with past latency measurements [7]. Dynamic metrics assume that network conditions fluctuate significantly, so they make metric measurements each time the client makes a request. A selection is then made from these fresh results.

Research has shown that dynamic metrics lead to more reliable selections than static and statistical metrics because network conditions demonstrate high variability. [2] shows that data transfers through the Internet vary drastically. [11] has observed that latencies and bandwidth across the WWW change throughout the day. [7] performed an empirical comparison between replica selection schemes and showed that dynamic schemes result in better selection than statistical schemes.

Selection schemes can also be categorized based on their point of selection. They fall into one of four categories: router, DNS, server-side, or client-side. In a router selection scheme, all the replicas serve under a common IP address. The routers select the best server for a client request [6]. DNS selection schemes utilize intelligent DNS servers that know the IP addresses of all the replicas for each domain name. When a client makes a request to a domain, the DNS resolves the domain name to the IP address of the replica that it calculates to be the best server [8]. Server-side selection schemes place the selection responsibility on the replicas themselves.

The types of metrics available to router, DNS, and server-side selection schemes are limited by their capabilities. Routers can only provide hop count measurements, which is generally insufficient as was mentioned previously. DNS selection schemes typically utilize a round robin scheme to make the selection. Other DNS selection schemes, such as the scheme used by Akamai Technologies, require a large infrastructure changes. Server-side selection schemes have easy access to information of server load. Yet, because the replicas are usually distributed across the WWW, server-side schemes have difficulty in accessing information about the paths between and client and each replica. In addition, server-side schemes require significant amounts of setup and implementation that may be undesirable or impossible since the replicas are remote over the WWW. Probe latency and path bandwidth – the two most reliable metrics for the WWW – are metrics based on path information, and server-side selection schemes cannot use these metrics. Client-side selection schemes are the only category that can measure these two metrics; therefore, they are the preferred schemes for the WWW.

We have so far concluded that dynamic, client-side selection schemes are best suited to our goals. [7] discusses three such schemes. The first scheme is called *dynamic probe*, where the client sends out a probe to all replicas after it sends out a request and selects the replica that first responds to the probe. This scheme uses probe latency as its selection metric. Results show that dynamic probe by itself offers a significant performance advantage over static and statistical methods. Two other schemes that incorporate bandwidth information are also presented. Both schemes use statistically calculated bandwidth as a tiebreaker for the dynamic probe scheme. These schemes send out dynamic probes upon each client request, but they do not select immediately after the first probe returns. They wait until a reasonable number of replicas have echoed the probe and select the server with the highest bandwidth among those that have responded. [7] concludes that two hybrid schemes do not offer improvement over the dynamic probe scheme. This result seems to contradict the intuition that bandwidth information is important in selection. We believe that the benefit of bandwidth does not show in these hybrid schemes because the bandwidth metrics used in the schemes are statistically collected beforehand and are not dynamic.

Application-layer Anycasting [3] is a client-side selection system that offers a suite of different selection schemes. Two of the schemes are variations of dynamic probe. It also offers a server-push scheme in which the server keeps track of its performance and occasionally pushes the information to the client. The client can then include server performance in making its selection metric. This performance data is more informative than server load and could be useful ([3] does not clearly validate it). However, to implement server-push requires server modifications; this requirement would limit the scalability and flexibility of the scheme.

Carter and Crovella propose a scheme that seems to provide the best client-side solution by combining probe latency and bandwidth [9]. It uses the concept of packet inter-arrival time to dynamically calculate path bandwidth and then mathematically combines it with probe latency to derive a metric. [9] proves that this metric accurately predicts the replicas' response times, but the metric is not implemented into the selection scheme because of the high time cost incurred by bandwidth measurement.

Two-phase probe uses a variation of the bandwidth measurement method used by Carter and Crovella [5]. Two-phase probe can utilize this method despite its associated costs because the measurements are made during the second phase of the scheme. The second phase occurs only after a client has already reached a replica of the web site through the first selection phase that uses dynamic probe. Based on the assumption of web user behavior made previously, the user is expected to make more than one file access from the same site. The second phase of two-phase probe occurs immediately after the user's first file access. We expect the user to spend at least a few seconds browsing the page before he requests another page. During that period of inactivity, two-phase probe has time to make the bandwidth measurements in the background, transparently to the user. By the time the user makes a second page access, two-phase probe can offer a more informative replica selection to the client. The details of how the two-phases cycle between each other are explained in Section III. It should be apparent now, however, that two-phase probe could make a selection incorporated with real-time bandwidth information without introducing a delay to the web site user's experience.

In summary, two-phase probe should improve upon the existing schemes described above because of the following:

- It is a dynamic selection scheme, so it selects replicas based on real-time network information. More accuracy can be achieved in the selection.
- It runs from the client-side. This gives it access to client-server path information. Further, it does not place requirements on the software of the replicas, which makes it flexible.
- It uses dynamic probe in the first phase, providing an initial selection in a very short time. This is done to provide initial service to the user quickly.
- It incorporates dynamic bandwidth information in the second phases, offering more accurate replica selections for subsequent file accesses. The time cost of bandwidth measurement is transparent to the user because it occurs during user inactivity.

3. Overview of Schemes

In our tests of two-phase probe, we compare the performance of two-phase probe against three other schemes: *round robin*, *dynamic probe*, and *dynamic bandwidth*. This section describes in detail each scheme with which we compare two-phase probe. It then explains two-phase probe, which combines dynamic probe with dynamic bandwidth.

Round Robin

Round Robin is the simplest selection scheme. It utilizes a list of the web site's replicas. The scheme remembers to which replica it selected for the previous request, and it always selects the next replica on the replica list for the next request. This scheme does not factor any network information into its selection and can be seen as a form of random selection. Its main benefit is even load distribution.

Dynamic Probe

Dynamic probe selects replicas based on the latency of a single echo datagram. Upon each client request to the web site, an echo datagram is sent to every replica server. The scheme selects the replica from which the client receives the first echo response.

Dynamic Bandwidth

Dynamic bandwidth is a much more complex scheme because it attempts to measure real-time bandwidth – a very difficult task. There are several tools to measure bandwidth such as CPROBE, PATHCHAR, TCPANALY, NETTIMER, and others [9] [10]. Each tool calculates bandwidth differently by measuring specific conditions in the network. We have chosen to use a modified version of CPROBE [5] as our bandwidth measurement tool. It is a dynamic measurement method that could estimate bandwidth purely through client-side operations. Our method estimates the bandwidth of the bottleneck link in a network path. It derives from the concept of inter-arrival times between packets that is discussed in [5]. In order to use this concept, we must make two assumptions. One assumption is that packets travel in immediate succession. Additionally, we assume they small packets are not reordered in the network path. In reality, there are exceptions to both assumptions; but we believe they are valid most of the time.

Under these assumptions, a client can theoretically measure path bandwidth by sending two echo packets with a short time gap between their sending. As the packets progress through the network to the server and back, they may encounter links that cannot process the packets as quickly as they entered them. When this happens, the gap between the two packets would be lengthened. Eventually, when the client has received both packets back from the server, the gap (or inter-arrival time) between the two packets should indicate the speed of the bottleneck link in the path they traversed (See Figure 1) from [5]. By dividing the number of bytes in the two packets by their inter-arrival time, one can arrive at the bottleneck bandwidth in the path. As long as the packets were initially sent with a gap that's smaller than what would be introduced by the bottleneck link, this calculation gives a valid estimate of bandwidth.

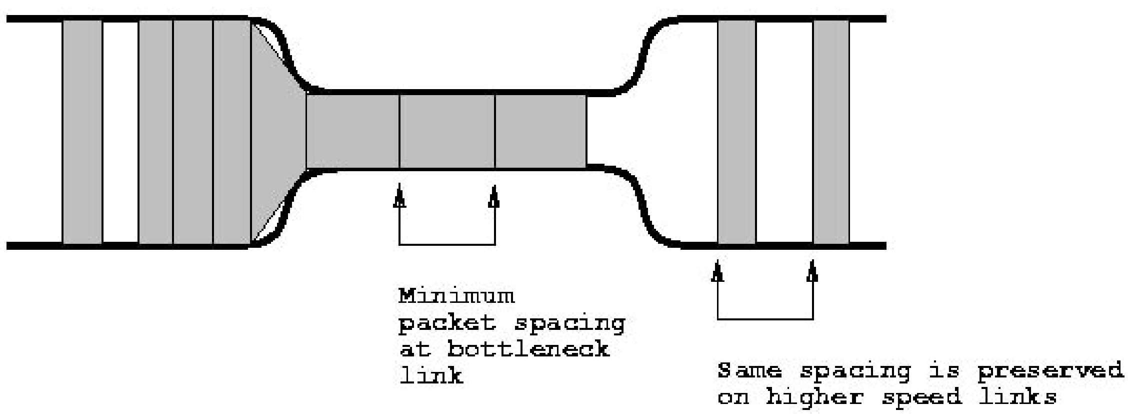


Figure 1. Packet spacing at bottleneck link.

Of course, the network exhibits enough variance that one set of inter-arrival time is insufficient to give a good assessment of bandwidth. CPROBE sends a sequence of 10 echo packets at once to the server. 1500 byte packets are used since they are some routers and links drop packets above that size. It is desirable to use a larger packet size in order to gain a more accurate spacing measurement. To avoid inaccurate results and to rule out overloaded servers, the servers that do not echo more than 50% of the sent packets within 300 milliseconds are eliminated from the selection pool. The bandwidth is calculated by dividing the total number of bytes sent by the inter-arrival time between the packets (see spacing in Figure 1). Dynamic bandwidth performs such a bandwidth calculation for each replica; and the replica with the highest bandwidth is selected as the "best" replica. We also adopt this bandwidth measurement method in phase II of two-phase probe.

Two-phase probe

Two-phase probe, as its name suggests, operates in two phases. We split the scheme into two phases to take advantage of our assumptions of web behavior stated earlier in the paper. The first phase uses dynamic probe for selection, and the second phase uses a scheme that combines dynamic probe with dynamic bandwidth. The goals of the two phases differ because the conditions in which they are used differ. In phase I, the user waits on the selection scheme, so the speed of selection is very important. Also, the client is expected to download smaller files during the phase. In small file downloads, server connection time and latency time dominate the response time, and therefore, dynamic probe provides an appropriate selection metric. In phase II, selection speed is not important because the user is inactive during the selection. Larger files may be downloaded in this phase, so dynamic bandwidth is incorporated to add information about transmission time to the metric.

Phase I of our selection scheme occurs when user accesses a web site for the first time in a while. As the user waits for the first web page to return, the client sends out a request waiting for the server's response. At this point, time spent on replica selection is noticeable to the user and becomes a part of the server response time. In the interest of minimizing selection time, dynamic probe is a good choice because it requires less time than other client-side schemes. Dynamic bandwidth information cannot be used in phase I because bandwidth measurement calls for a time cost that is at least a few seconds. Such a large time cost is unacceptable because it would undoubtedly increase the effective server response time. Since we assume that the first web page a web user retrieves does not entail large file transfers, bandwidth information is not an extremely relevant issue anyway. Dynamic probe presents a good balance between short selection time and high selection accuracy. After phase I, the client is directed to a replica selected by dynamic probe.

The next phase of selection happens as soon as the client has finished receiving the response to its first request. At this time, we expect the user to remain inactive for at least a few seconds. During this period of inactivity, measurements of path bandwidths are made (as mentioned above in *Dynamic Bandwidth*). Since the user is usually inactive immediately after the first request, the bandwidth measurement process occurs transparently to the user.

Phase II of two-phase probe also takes into account latency in its selection. Besides calculating bandwidth, phase II also logs the latencies of the probe packets echoed from each replica. The mean of the probe latencies for each replica is calculated. At this point, all the replicas are ranked by their mean probe latencies – replicas with lower mean latencies are ranked better (i.e. given a lower rank number). To include the collected bandwidth information, the replicas are then ranked by bandwidth – replicas with higher bandwidth are ranked better. Now that the scheme has knowledge of latency and bandwidth for each replica, it attempts to find the replica that exhibits the ideal balance between the two metrics. It does so by adding the latency and bandwidth rank numbers for each replica to arrive at a combined ranking index. For example, if a replica is ranked 3^{rd} in latency and 2^{nd} in bandwidth, its ranking index would be 5. Phase II selects the replica with the lowest ranking index. This scheme gives latency and bandwidth equal weight in selection. Since it is difficult to know whether a user needs high bandwidth or low latency [1], giving equal weight to each metric is a fair compromise.

The replica chosen by phase II is cached for a 60 second period. The next client request to the same site is directed to the cached replica. When the cache expires or when the client accesses another web site, two-phase probe returns to phase I for replica selection. The two-phase cycle thus begins again.

4. Testing Methodology

In our tests of two-phase probe, we attempt to answer three questions:

- Does two-phase probe offer better performance than the benchmarks?
- Does Internet traffic affect selection performance?
- Does the size of the file requested influence selection performance?

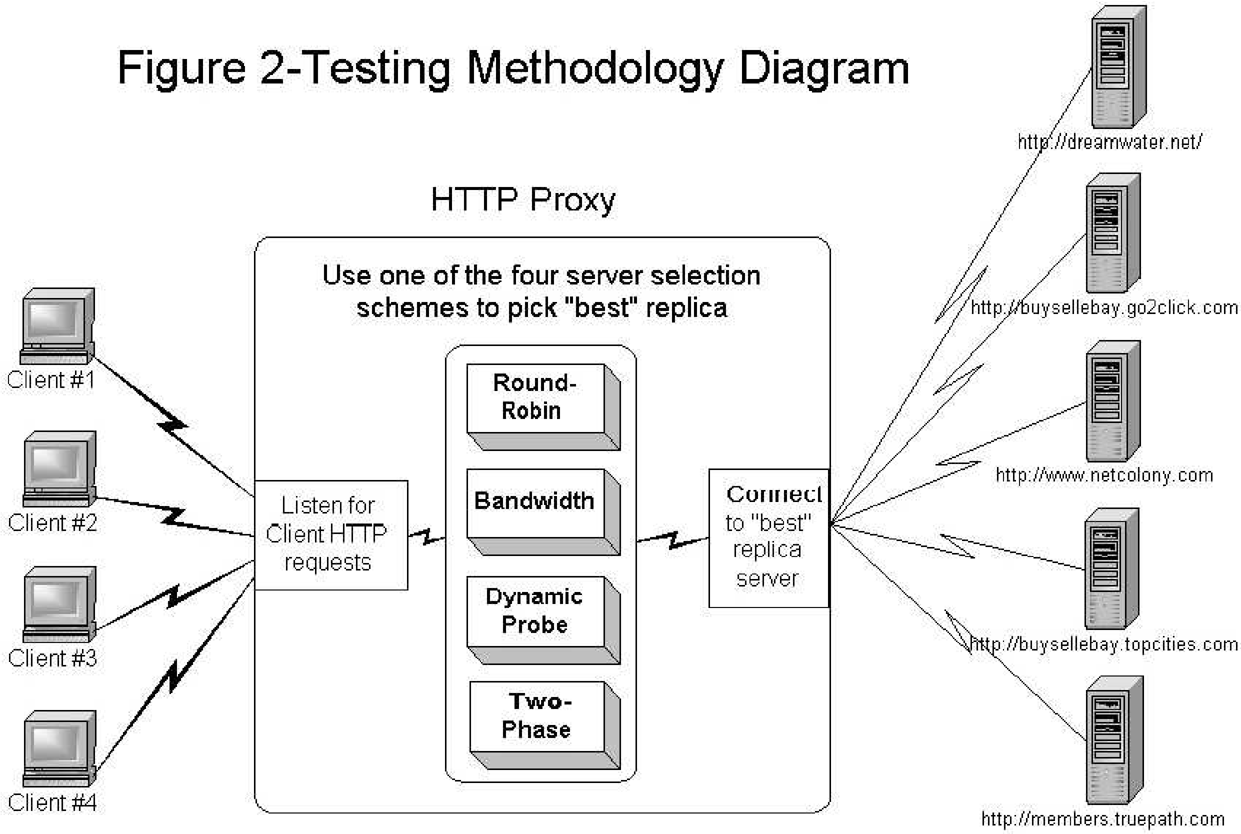
Our test utilizes five web servers distributed around the nation as replica servers:

- www.dreamwater.net
  Location: Alexandria, VA *Server*: Apache/1.3.11 (Unix)
- www.go2click.com
  Location: San Jose, CA *Server:* Apache/1.3.6 (Unix)
- www.netcolony.com
  Location: Davie, FL *Server:* Apache/1.3.12 (Red Hat/Linux)
- www.topcities.com
  Location: Buena Park, CA *Server:* Apache/1.3.12 (Red Hat/Linux)
- www.truepath.com
  Location: Englewood, CO *Server:* Apache/1.3.12 (Unix)

We intentionally chose web servers that are geographically spaced apart to simulate a real national distribution of replicas. To each of these web sites, we upload three files of varying sizes: 3K, 30K, and 100K. The IP address of each website is maintained on the client testing application.

Four UltraSPARC 10 workstations in the same MIT subnet running the Solaris 7 operating system are used as testing clients. We install a different testing suite on each machine, one for each of the four following selection schemes: round robin, dynamic probe, dynamic bandwidth, and two-phase probe. Each suite consists of two parts: a proxy and a testing client. The client's purpose is to send file requests, and the proxy's responsible for selecting the replica and logging the statistics. All testing suites share the same testing client, but the proxies differ slightly because of their different associated selection schemes.

Figure 2-Testing Methodology Diagram

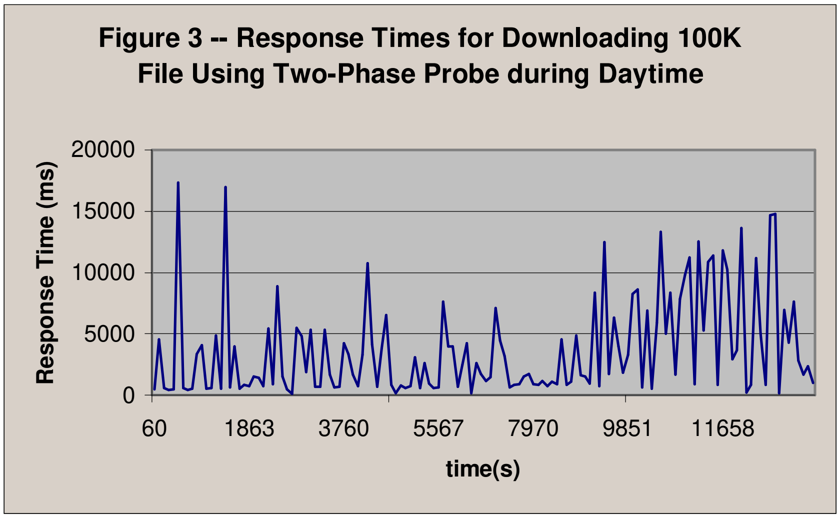


A testing session begins by simultaneously starting the testing clients on all four clients. Each client thus begins a cycle of file downloading and statistics logging. Every 30 seconds, the client sends a request to download a file, cycling between the 3K, 30K and 100K files in round robin fashion. The constant variation in file size provides a better simulation of reality than the repeated download of one file. The proxy intercepts the HTTP requests sent by the testing client and directs them to the best replica selected by its associated selection scheme. Figure 2 depicts the testing setup.

For each request, the time sent, file requested, and server response time are logged by the proxy. We run two separate testing sessions of three hours each – one during the day when internet traffic is heavy and one during the night when traffic is light. Overall, we collect statistics of roughly 350 requests per client per session.

5. Results

We compare the median server response times of the four schemes for our daytime and nighttime sessions. Before looking at the comparison results, the motivation behind using the median of the response times should be discussed. Initially, we decided that the mean response times of the schemes were compared instead, but they exhibited very inconsistent values. We examined the statistics from the testing sessions more closely and noticed drastic fluctuations in server response times – a few values were up to 10 times higher than the mean! Figure 3, which shows the server response time of two-phase probe in downloading the 100K file during daytime, demonstrates this phenomenon. There are repeated wild variations in the response times that indicate the inconsistency of network conditions. Some of the extremely large outliers greatly bias the mean, and therefore, the median response time is a more reliable measure of typical server response times.



The median response times of the four selection schemes are summarized in Figures 4 and 5. It is important to note that the time overhead of bandwidth measurement is not included in the server response statistics for dynamic probe, so we can purely observe the predictive ability of the bandwidth information. The x-axis for each figure shows the size of the file requested by the client and the y-axis represents the median response time to retrieve a file for the specified selection scheme. These graphs show that both dynamic probe and two-phase scheme can provide a large reduction in server response time over round robin. These results are expected, since dynamic probe and two-phase probe incorporate real-time path information while round robin uses no intelligence in its selection. The results of dynamic bandwidth are more surprising. We do not expect bandwidth, by itself, to be an accurate predictor of server response time, but we do expect it to fare better than round robin. Apparently, that is not the case. In fact, it performs so similarly to round robin that we conjecture that dynamic bandwidth provides no useful information.

Figure 4 -- Median Response Times of the Four Selection Schemes (daytime)

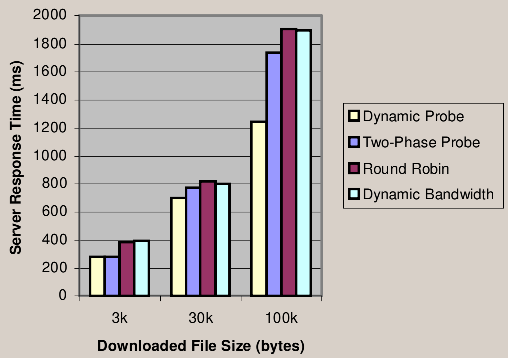
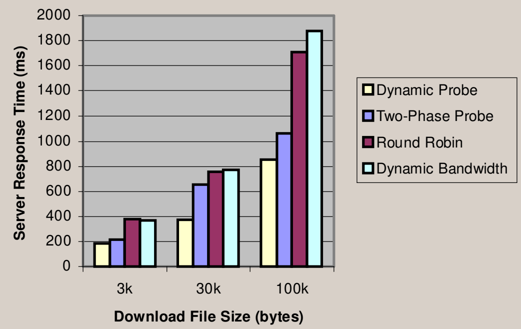


Figure 5 -- Median Response Times of the Four Selection Schemes (nighttime)



Looking at Figures 4 and 5, the dynamic probe scheme consistently provides the best median response times for all file sizes during both periods of the day. The two-phase selection scheme provides the second best response times. The round robin and bandwidth selection scheme appear to have similar performance. These relative rankings stayed the same across daytime, nighttime, and across different file sizes.

From these results, three inferences can be made. First, latency predicts server response time accurately, judging from the superior performance of dynamic probe. In times when there is not much network traffic—like during the nighttime—using latency as the sole criteria for server selection reduces response times by up to 50% when compared to other selection schemes. Latency is a good metric for optimizing the total response time especially during times of low network traffic.

By observing the performance of the bandwidth selection scheme, we can make one of two inferences. Figures 4 and 5 indicate that dynamic bandwidth performs similarly to round robin. The log files generated from the test sessions show that dynamic bandwidth selects each of the five replicas almost with equal frequency. This points to the conclusion that dynamic bandwidth selects servers almost randomly. One possible explanation for this phenomenon is that bandwidth provides no predictive information to server response time. This seems unlikely, since bandwidth has such large influence in file transmission time, which is a large part of server response time. Another explanation is that the bandwidth measurement method we chose, a customized version of CPROBE, is inaccurate. This is a more likely explanation, since the difficulty of measuring Internet bandwidth has been well documented by current research.

Since our bandwidth measurement method seems to provide random information, we can infer that the resulting bandwidth measurements hurt the accuracy of two-phase probe. Phase II factors latency and bandwidth equally in its selection, so the randomness of bandwidth information drags two-phase probe's performance to a level between that of round robin and dynamic probe. These results do not reveal the actual performance of two-phase probe. It is possible that two-phase probe may perform better than dynamic probe if an accurate bandwidth measurement method is used.

6. Conclusion

This paper evaluates the server selection schemes proposed in [7] and a hybrid selection scheme that incorporates both dynamic latency and dynamic bandwidth measures. The hybrid, which we call the two-phase probe selection scheme, meets three of our four design goals: minimal observable delay to the user, transparency to the user, and minimal implementation requirements. It introduces almost no observable delay to the user. During Phase I, dynamic probe, a scheme that introduces almost no selection delay, is used for selection. In Phase II, the selection process consumes up to seconds of time, but the user cannot observe it because it occurs when the user's inactive. It is transparent to the user because it requires no input from the user for selection. The short selection time overhead also contributes to transparency. The architecture requires no changes to the infrastructure or any of the replica servers, which makes it ideal for the heterogeneous, fast-changing WWW.

Unfortunately, the primary design goal of the two-phase probe—to provide the lowest response time to the client's request—is not achieved in our implementation of the scheme. The dynamic probe selection scheme clearly achieves lower server response times than other schemes; even though theoretically, we believe that two-phase probe should perform at least as well, if not better than, dynamic probe because it incorporates a bandwidth metric. We accredit this to the variability in network conditions. In reality, bandwidth fluctuates so much that it is difficult to obtain stable and reliable measurements of it. The imprecise bandwidth measurements are the most probable reason behind two-phase probe's weak performance.

Bandwidth has such a large influence on file transmission time, we believe that an ideal replica selection scheme must include a bandwidth metric. Both [9] and [10] explain various bandwidth-measuring techniques. However, the methods other than CPROBE require either a large amount of overhead or particular modifications to the servers; both requirements conflict with our design goals. Two-phase probe is a solution we propose that incorporates bandwidth, but an efficient and reliable method for measuring bandwidth needs to be developed before the scheme's potential can be accurately evaluated.

References

[1] Arlitt M. F., Williamson C. L., “Web Server Workload Characterization: The Search for Invariants”, ACM SIGMETRICS, May 1996

[2] Balakrishnan H., Seshan S., Stemm M., and Katz R.H., “Analyzing stability in wide-area netowork performance,” in *Proc. of ACM SIGMETRICS Conf. on Meas. & Modeling of Computer Systems,* June, 1997.

[3] Bhattacharjee S., Ammar M. H. Zegura E. W., Shah V., Zongming F., “Application-Layer Anycasting”, Proceedings of the IEEE, Volume: 3, May 1997

[4] Carter R. L., Crovella M. E., “Dynamic Server Selection in the Internet”, Third IEEE Workshop, 1995.

[5] Carter R. L., Crovella M. E., “Measuring Bottleneck Link Speed in Packet-Switched Networks”, BU-CS-96-006, Boston University, 1996.

[6] Cisco Distributed Director. http://www.cisco.com.

[7] Dykes S. G., Robbins K. A., Jeffrey C. L., “An Empirical Evaluation of Client-side Server Selection Algorithms”, IEEE Infocom, May 2000.

[8] Katz E.D., Butler M., and McGrath R., “A scalable HTTP server: The NCSA prototype,” *Computer Networks and ISDN Syst.*, vol. 27, pp. 155-164, 1994.

[9] Lai K., Baker M., “Measuring Bandwidth”, Proceedings of the IEEE, Volume: 1, 1999.

[10] Lai K., Baker M., “Measuring Link Bandwidths Using a Deterministic Model of Packet Delay”, Proceedings of ACM SIGCOMM 2000*,* August 2000.

[11] Mogul J., “Network behavior of a busy web server and its clients,” *Tech. Rep. Research Report 95/5,* Digital Equipment Corporation, Oct. 1995.