In Proceedings of IEEE Infocom '97, April 1997, Kobe, Japan

Server Selection using Dynamic Path Characterization in Wide-Area Networks

Robert L. Carter and Mark E. Crovella
Computer Science Department
Boston University
Boston MA 02215

December 5, 1996

Abstract

Replication is a commonly proposed solution to problems of scale associated with distributed services. However, when a service is replicated, each client must be assigned a server. Prior work has generally assumed that assignment to be static. In contrast, we propose dynamic server selection, *and show that it enables* application-level congestion avoidance.

Using tools to measure available bandwidth and round trip latency (RTT), we demonstrate dynamic server selection and compare it to previous static approaches. We show that because of the variability of paths in the Internet, dynamic server selection consistently outperforms static policies, reducing response times by as much as 50%.

However, we also must adopt a systems perspective and consider the impact of the measurement method on the network. Therefore, we look at alternative low-cost approximations and find that the careful measurements provided by our tools can be closely approximated by much lighter-weight measurements. We propose a protocol using this method which is limited to at most a 1% increase in network traffic but which often costs much less in practice.

Keywords: Distributed Systems, Internet and Network Algorithms

1 Introduction

The increasing popularity of distributed information services like the World Wide Web has resulted in a number of problems of scale. Three scaling impediments to such distributed information services are excessive server load due to document popularity, wasted network bandwidth due to redundant document transfer, and excessive latency in delivering documents to the client due to the potential for transfers over slow paths. While caching can alleviate the problem of repeated transfer, a technique that promises to address the remaining problems is service (or document) replication. However, when a service is replicated, clients face the additional task of finding the best provider of that service.

In many cases, clients may know in advance which service providers are best for them. Such a static server selection scheme is used, *e.g.,* in the distribution of network news using NNTP. However, static server selection is inappropriate in a number of cases. For example: 1) The authors in [5] employ a dynamic protocol to find the best server for a document: the client probes a set of servers who may have a replica of the document, as well as the document's home server, and chooses the first to reply as the source of the document. 2) A mobile client using a replicated service will want to find the best server to serve the client's requests; the choice will depend on the changing location of the client. 3) Finally, our current interest is in replicating WWW documents. In this context, dynamic server selection has the potential to reduce the response time – the elapsed time between document request and document arrival. In addition, as we will show, a dynamic server selection mechanism enables a very flexible, simple policy for document replication.

In this paper we report on tools and techniques for selecting good service providers without assuming knowledge of server location or network topology. Our only assumption is that the client can obtain a list of addresses of servers that provide the required service.

In our initial experiments we consider two principal metrics for measuring distance in the Internet — hops, and round-trip latency — and study their use as the basis for a server selection decision. Surprisingly, we show that these two metrics yield very different results in practice. We then present evidence that dynamic server selection polices based on instantaneous measurements of round-trip latency provide considerable reduction in response time compared to static policies (which in this case are based on either geographical location of client and server or distance measured using network hops).

However, round-trip latency alone does not capture all the information one would want about the quality of a connection. In the context of server selection, another important characteristic of a network connection is the bandwidth available to clients of that connection. All other things being equal, higher available bandwidth implies faster document transfer time. Available bandwidth depends on two things: 1) the underlying capacity of the path between client and server which is limited by the slowest (or *bottleneck*) link, and 2) the presence of competing traffic (*congestion*).

These two useful pieces of information are not readily available to applications. In order to discover this information we developed two tools: BPROBE, which measures the uncongested bandwidth of the bottleneck link of a connection; and CPROBE, which estimates the current congestion along the bottleneck link of the path. The design and validation of these tools is described in [4]. In the remainder of this work, we use CPROBE and its available bandwidth measurements.

Armed with measurements of the current network state we can perform application-level congestion avoidance. By application-level congestion avoidance we mean that applications can use estimates of traffic volume to actively avoid paths that are currently heavily loaded. Specifically, in this paper we examine how a combination of measurements of latency and available bandwidth can be used to formulate a dynamic algorithm for server selection. We show that the combination of both measurements improves our dynamic server selection policy over any one metric alone.

The current load at the server is also an important factor when choosing a server. However, it is difficult to assess without system software modifications at servers. In [3], we do show the results of a simple experiment using a surrogate metric for server load, and show evidence of the importance of including a measure of server load in dynamic server selection. Our results suggest that a low-cost method of measuring server load would be useful and we plan to pursue this issue.

Our methods can be viewed from two standpoints: from a client-oriented standpoint, they reduce response time, which is good; from a system-oriented standpoint, they add traffic to the network, which is bad. In the first part of the paper we concentrate on showing that response time can be radically reduced using (sometimes expensive) measurements. In the latter part of the paper we show how almost all client-oriented benefits can be obtained using much less expensive, but approximate, measurements. Our final scheme obeys a strict limit of not increasing overall traffic in the network by more than 1%.

In the remainder of the paper we describe our proposed solution to the server selection problem and evaluate it in terms of both user-oriented and system-oriented costs and benefits. First, we divide the universe of documents into two classes: small and large documents. We expect the transfer time for small documents to be dominated by RTT, so we use simple, low overhead, RTT measurements as a basis for dynamic server selection. Then, for larger documents, where we expect bandwidth limitations will dominate the transfer time, we use a combination of RTT measurements and measurements of available bandwidth derived from our CPROBE tool. After considering the costs of various measurement approaches we show that simpler measures can be as effective while costing less both in terms of measurement time and network bandwidth overhead. We then present our limited overhead dynamic server selection protocol suitable for documents of any size and show its improved performance compared to static policies and minimal impact on network bandwidth. We conclude with a discussion of ongoing and future work.

2 Related Work

In [8], a replication technique (and presumed server assignment) based on geographical distance is proposed. In their design, servers are responsible for placing replicas near sources of high demand. Clients request the address of the nearest replica from the home server which calculates distances in miles to find the replica closest to the requesting client. This is essentially a static method of server selection and relies on the server maintaining a large amount of geographical information. In contrast, we show below that a dynamic server selection policy can provide high performance and a simplified placement policy.

Distance measures are also the focus of [7] where the objectives are to keep communication local and limit the use of long-haul links. The authors consider that the high variability of round-trip times renders them unattractive in practice. Conversely, we show this high variability is an essential feature that can be exploited when selecting a server. Instead of using hops or round-trip time, all of the methods explored in that work attempt to determine a subset of the network topology and build a distance metric on that topology. Hence, the result is again a static policy. In contrast, our dynamic server selection polices do not require explicit knowledge of topology. Furthermore, as we will show, the quickly changing and highly variable conditions in the Internet require a degree of dynamic assessment that static policies do not provide.

A dynamic server selection policy based on measured latency was used to choose among hierarchically organized caches in Harvest [5]. The cache's resolution protocol performs a remote procedure call to all of the siblings and parents of the cache and checks for hits. The cache then retrieves the object from the responding site with the smallest measured latency. This is similar to the policy we call Dyn 1 below. In this paper, we show that the additional time required for more complete measurement of current network conditions often results in improved performance.

3 Dynamic Server Selection

3.1 Why Dynamic Server Selection?

In any server selection scheme, a client seeking a document would like to choose the server which it has reason to expect will deliver the document in the shortest amount of time. In this section, we show that the impact of this choice is strongly affected by whether it is made based on static assignment or based on dynamic measurements of current conditions.

The difference between dynamic and static approaches to server selection is illustrated by the measurement of distance. Two obvious metrics for measuring distance in the Internet are hops, and round-trip latency. However, the static nature of the hops metric (the number of hops between any two hosts rarely changes) is considerably different from the quickly changing, widely varying RTT metric. These differences are important in solving the server selection problem.

We begin by comparing empirically measured distributions of each metric. We measured the values of both metrics between a fixed client host and 5262 servers selected randomly from a list of WWW servers [9].

Figure 1 shows on the top, the measured distribution of hops to the set of servers; and on the bottom, the distribution of round-trip latencies (in ms, measured using *ping*). These distributions are strikingly different. The distribution of hops appears to be fairly symmetric about its mean, whereas the distribution of latencies has a median (125) much less than the mean (241). The differences between the distributions suggest that hops is a poor predictor of latency.

The fact that hops is not a good predictor of latency suggests that either variation in link speed or delays due to congestion dominate the round trip time of packets. The impact of congestion is made clear by examining time series plots of round trip times to a single host. Unsurprisingly, these time series plots show extreme variance over short time periods. Consider Figure 2, which presents measurements of latency to a single host gathered over a period of two days at intervals of 30 seconds. On the top is a time-series plot; on the bottom a histogram of the same data. The variation in latency measurements reflects the underlying changes in congestion. A dynamic server selection policy can be designed to take advantage of exactly this sort of variation.

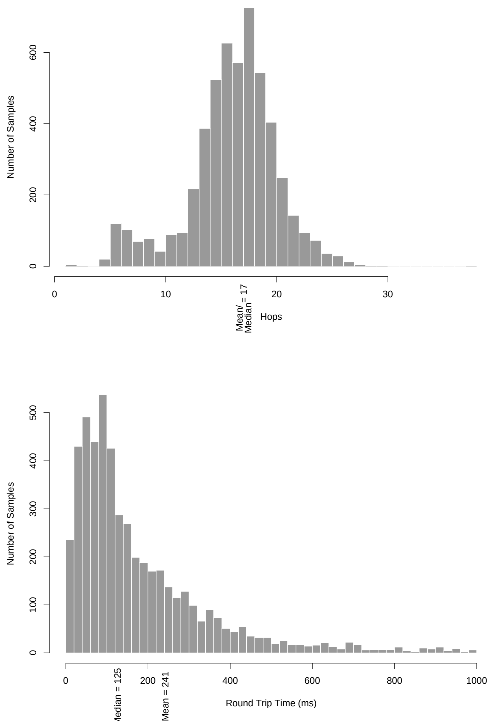

Figure 1: Empirical Distribution of Hops and RTT to 5262 Random Servers

In fact, the difference between the characteristics of hops and latencies is fundamental enough to also suggest differences in algorithms for server replication. An initial replica placement policy might try to place replicas "close" to the clients that will use them (*e.g.,* [8]). This follows naturally when thinking about hops as the distance metric. Because the bulk of the hops distribution is centered about the mean, care is re-

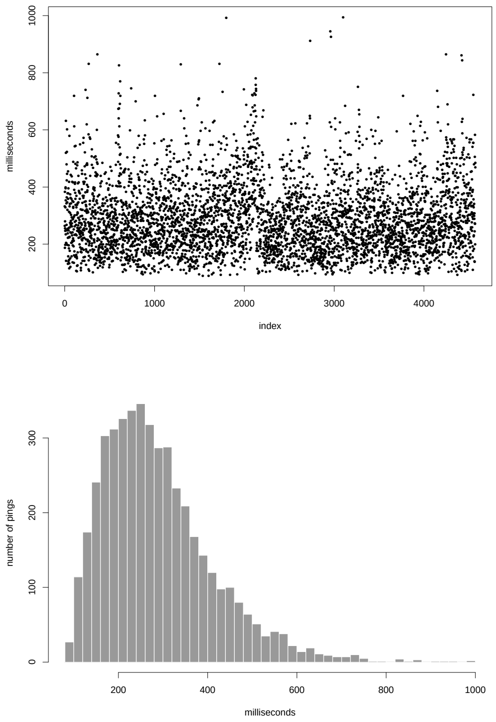

Figure 2: Round Trip Times to a Single Host

quired in placing replicas if the goal is to minimize the number of hops between client and server. In other words, a random distribution of replicas will not significantly change the mean distance in hops between clients and the servers they use. On the other hand, this is clearly not the case when the measure is round-trip latency. Because the bulk of the probability mass of the latency distribution lies below the mean, a random placement strategy should markedly reduce the mean latency between client and server.

Taken together these observations indicate that the performance of a replicated server system that uses dynamic server selection based on round-trip latency will be less sensitive to replica placement than a system that relies on the static approach where hops is the measure of distance.

3.2 Server Selection for Small Files - RTT

Having established the theoretical benefit of dynamic server selection, in this section we present experimental results to show that even very simple dynamic server selection policies can outperform static approaches in the real Internet. As previously stated, our dynamic policy operates under the assumption of widespread replication of documents (such as proposed in [1, 6]). We further assume that it is possible for a client to obtain a list of hosts serving replicas of the document of interest (perhaps by contacting the home server of a document).

In order to simulate such a system, we identified 10 hosts with approximately equal average round-trip latency as measured from our test client. Then, over a 3-day period, for each host we measured (about once per hour) the round-trip latency using *ping*, and the transfer time for documents of sizes 1KB, 5KB, 10KB and 20KB. We chose these smaller file sizes as appropriate to test the simple RTT-based server selection methods since latency should be the dominant factor for small transfers.

Using this data we then simulated several server selection policies: 1) Static, based on minimizing geographic distance; 2) Static, based on minimizing the number of network hops between client and server; 3) Dynamic, based on random selection of a server; and 4) Dynamic, based on minimizing the mean of 1,2,3,4 or 5 round-trip measurements. The results are summarized in Figure 3. For comparison, we also include Optimal and Pessimal polices which are simply the minimum and maximum transfer times observed, respectively. The graph shows the average time to fetch an object for each of the 4 file sizes. For the policies based on RTT measurements, the time plotted includes the time to make the required number of RTT measurements. Static policies are presumed to have no additional cost since the preferred server can be found by consulting a table.

All of these policies were evaluated with the objective of minimizing user response time. In this respect, as can be seen in Figure 3, the dynamic policies consistently outperformed the static policies and the benefit of a dynamic policy generally increased with document size. Even random selection is preferable to static polices for documents larger than 5KB. In this simulation, the dynamic policy based on the mean of 4 round-trip time measurements, gives the best results, minimizing the transfer time (inclusive of measurement time). Dynamic policies based on fewer (1,2 or 3) RTT measurements do a little worse. A policy using 5 RTTs does worse still, exhibiting a phenomenon of diminishing returns for extra measurement overhead.

The best dynamic policy in this simulation required an average transfer time that was only 4 times the optimal for 5KB files and this improved to less than 2 times optimal for larger files. In contrast, the static

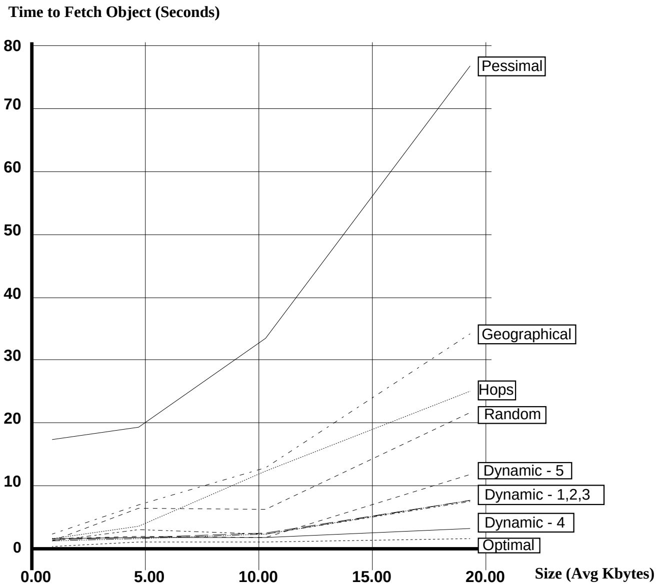

Figure 3: Fetch Times of Static and Dynamic Policies Plotted Against Document Sizes.

hops policy which was also about 4 times optimal for small files, showed increasingly worse performance as transfer size increased, up to 20 times optimal for 20KB files. These results use 7 replicas, certainly not a very large number for WWW document replication. The results would surely improve with a greater degree of replication.

However, as seen in Figure 3, the benefit of multiple measurements of round-trip time as a predictor of transfer time relative to the optimal value decreases with increasing file size. We believe that this indicates the effect of limited available bandwidth due to competing traffic, which will have a greater impact on larger document transfers, while small document transfers would have their transfer time dominated by instantaneous latency. This observation led us to formulate the techniques mentioned earlier for measuring the bottleneck link speed and available bandwidth of a path. Server load may also be a contributing factor, but as we discuss further below, the current lack of a simple way of measuring server load prevented us from studying this directly. We next discuss dynamic server selection using measurements of available bandwidth in addition to latency and its application to larger service times.

3.3 Server Selection for Large Files - PTT

The results of the previous section clearly show the benefit of dynamic policies. However, the relative benefit was observed to decrease with increasing transfer size, a portion of this effect was hypothesized to be due to bandwidth limitations. In this section we present a server selection policy, PTT, appropriate for larger files, based on dynamic bandwidth measurements. We will use the term *available bandwidth* to refer to the estimated transfer rate available to the application at any instant – that is, the portion of raw bandwidth which is not currently used by competing traffic.

We suspected that one of the factors involved in the reduction in predictive ability of the RTT policies as document size increased was limited available bandwidth. In order to estimate available bandwidth, we developed the CPROBE tool mentioned above (and presented in detail in [4]) . In this section we study the use of bandwidth measures concentrating on documents larger than those considered in Section 3.2. In the context of server selection for WWW documents, given these measurements and the size of the document to be retrieved, it is possible to estimate the transfer time directly. If the document size is not known, then a choice could be made based on, for example, highest available bandwidth.

Given the bandwidth probing tools in addition to the RTT estimators commonly available, two questions arise: 1) can we improve the predictive capability of our dynamic server selection algorithm? and 2) what is the additional cost of the probes?

3.3.1 Evaluating Detailed Server Selection

In order to evaluate server selection policies based on available bandwidth measurements, we collected data from several WWW servers in the Internet. From earlier work [2] we have available a database of document size and location information. From this source we extracted sets of documents of sizes 100KB, 500KB, 750KB and 1MB. We chose these larger sizes because the transfer times of these large documents should be influenced by available bandwidth as well as latency. Periodically, over a several hour period we recorded the following data for each document: 5 round-trip time measurements (using *ping*), the available bandwidth (using CPROBE) and the transfer time for the document.

Before beginning simulation of the various selection policies, we used linear regression to analyze the dependence of measured transfer time on: 1) the RTT as measured by *ping*; and 2) the predicted transfer time using a combination of RTT and available bandwidth. Table 1 gives the R^2 values of the regressions. The regression analysis shows an improvement in accuracy of predicted transfer time when the measurements from the bandwidth probing tools are added to latency measurement provided by a single ping. Also the predictive value of our measure increases with larger document size, except for the very largest documents. We suspect that the long transfer times associated with the largest documents extend beyond a valid prediction window. As we have shown above, the effect of competing traffic on available bandwidth is a highly variable one. Therefore, estimates have a limited useful lifetime; we are currently exploring methods for assessing those limits.

	Document Sizes			
Regression Formula	100K	500K	750K	1MB
xfer time vs. single *ping*	0.053	0.186	0.477	0.081
xfer time vs. *ping* and CPROBE	0.291	0.280	0.638	0.145

Table 1: Regression Analysis: R^2 values for two metrics for 4 document sizes

To use *ping* and CPROBE together, we formulate the predicted transfer time metric. We use the regression coefficients k_1 and k_2 which relate the predicted transfer time to round-trip latency and a bandwidth-limited component, as follows:

$$\text{PredictedTransferTime} = k_1 \text{RTT} + k_2 \frac{\text{document size}}{\mathcal{B}_{avail}},$$

where $\mathcal{B}_{avail}$ is the available bandwidth as measured by CPROBE. In the simulation of this selection policy (PTT), the predicted transfer time is calculated for all candidate servers and the server with the minimum value is chosen.

We again simulated both static and dynamic server selection algorithms using the data collected from our survey of WWW servers. For each document size (100K, 500K, 750K and 1MB) we selected 7 data points uniformly from the recorded data and applied the dynamic algorithm based on RTT and available bandwidth to make a selection from among the 7 sites. The results are presented in Table 2. Each entry in the table represents the mean transfer time in seconds over 1000 simulated server selection decisions: that is, transfers of documents of the size given by the column heading when the server is selected according to a policy given by the row heading.

The policies simulated were: 1) Pessimal, which simply chooses the worst (largest) transfer time among the 7 servers; 2) Random, which picks a server uniformly; 3) Hops, which chooses the server which is the fewest hops away; 4) PTT, as defined above; and 5) Optimal, which chooses the best (smallest) transfer time.

	Transfer Sizes			
Selection Policy	100K	500K	750K	1MB
Pessimal	21.836	21.800	18.357	696.680
Random	6.857	7.989	10.572	281.919
Hops	2.267	8.958	10.120	46.190
PTT	2.712	2.001	5.721	31.897
Optimal	0.985	1.296	4.265	17.529

Table 2: Simulation results for large transfers - mean transfer time in seconds

The superiority of the dynamic policy for server selection is clear from Table 2. The improvement is especially marked for large documents. For example, for 500KB documents, an improvement of over 75% is found when using PTT rather than Hops (2 seconds versus nearly 9 seconds). The transfer time chosen by the dynamic policy is less than 2 times the Optimal time for large documents, while the Hops policy results in transfers taking at least $2\frac{1}{2}$ times the Optimal time.

4 Cost Considerations

In addition to time overhead for network probing, which is reflected in the response time measurements we present, the impact on the network is also a critical issue. Probes that add many extra bytes of network load certainly will be unwelcome.

The current implementation of CPROBE uses 4 sets of 10 packets each with a maximum timeout of 1 second per set. Packet sizes of 600, 700, 800 and 900 bytes are used for a total overhead of 30,000 bytes and up to 4 seconds of measurement time. So, the probe may add up to an additional 30,000 bytes to the network load and a few extra seconds for an accurate measurement of available bandwidth. Such costs are too high for widespread use.

To address this, we explored measurement methods that can have much lower network impact and yet would approximate the more expensive CPROBE. In the next section we present such methods, and then propose a policy based on these lower-cost measurements which limits probe overhead to no more than 1% additional traffic in the network.

	Transfer Sizes			
Selection Policy	100K	500K	750K	1MB
Hops	2.267	8.958	10.120	46.190
Dyn 1	2.124	2.147	6.612	50.857
Dyn 5	2.301	2.028	5.762	31.175
PTT	2.712	2.001	5.721	31.897

Table 3: Simulation results for large transfers - mean transfer time in seconds

4.1 Approximate approaches to PTT

Previously we showed that, in theory, the bandwidth measurements combined with latency (RTT) measurements have greater predictive capability than latency alone. However, we suspected that a less expensive policy based the mean of more than one RTT measurement, might provide a rough approximation to a bandwidth estimation since long RTTs may be indicative of low available bandwidth. First, we compare the performance of the combined policy (PTT) with the simpler RTT-only policies. Then we address the issue of cost of the additional measurements.

In Table 3 we add two more dynamic policies: 1) Dyn 1, which performs a *ping* to each server and chooses the server with minimum RTT; and 2) Dyn 5, which computes the mean of 5 *ping* measurements and chooses the server with the minimum value. Both of which were also used in the initial study described in section 3.2. Once again, this table shows that a dynamic policy is always superior to the static policy based on distance measured using hops, confirming our earlier results. Also, for large files, the use of the probe measurements improves over the results from a single latency measurement.

The surprising result is that the Dyn 5 policy which relies solely on RTT measurements gives results very close to those of PTT which uses the additional measurement of available bandwidth. A similar though weaker correlation persists when using fewer than 5 RTT measurements. Table 4 gives values for the correlation coefficient between the reciprocal of the available bandwidth (measured using CPROBE) and the average of n RTT measurements for $n = 1, 2, 3, 4, 5$. Clearly the correlation increases with the number of RTT measurements, implying a strong link between the number of RTT measurements made and the degree to which those measurements capture the available bandwidth. As shown in the graph, more RTT measurements give an estimate with greater correlation to the available bandwidth.

Number of pings				
1	2	3	4	5
0.236	0.293	0.346	0.372	0.400

Table 4: Correlation between available bandwidth measurement and RTT measurement (average of 1,2,3,4 or 5 RTT measurements).

To summarize, both Table 3 and Table 4 suggest that the policy of minimizing the mean of 5 *ping* measurements already accounts in a way for congestion effects. In other words, direct measurement of congestion as we have done with CPROBE can by simulated to some degree by using a sequence of RTT measurements. This is significant because of the (currently) high overhead of the probe measurements. These observations suggest a limited overhead dynamic server selection policy introduced in the next section.

4.2 The OnePercent Protocol

Since the performance of the lighter-weight probes closely approximates the more expensive ones, we devised the following probing protocol. Paying particular attention to impact on the network, we propose a protocol that permits at most a one percent overhead in terms of additional bytes injected into the network. We therefore refer to this policy as OnePercent. That is, for each document request, the number of additional bytes used to probe the network is limited to at most one percent of the size of the document requested. This protocol also ensures that the additional time needed to probe the network state is proportional to the document size. Thus, larger documents, which will benefit from more precise measurements of network conditions, are provided with better measurements while smaller documents for which latency is the dominant factor will not increase response time or use network bandwidth unnecessarily.

The OnePercent policy is formulated as follows: for files under 10,000 bytes, we use a single ping; under 20,000 bytes, we use the mean of two pings, etc. Because the CPROBE overhead is 30,000 bytes, all of the larger documents will use the mean of 5 pings in the OnePercent protocol. Thus, we use at most one ping per 10K bytes to be transferred.

Figure 4 presents the results of applying the OnePercent protocol. For each document size, there are 5 bars showing the response (transfer) time in seconds for the best and worst transfer times (Optimal and Pessimal, respectively); for Random selection of server; for static selection based on minimizing number of Hops; and for the OnePercent dynamic probing policy. In order to make visible the values for smaller documents, the bar chart was cut off at 100 seconds (the 1MB Pessimal time is 700 seconds, the 1MB Random time is 281 seconds). From the figure, it is clear that the OnePercent policy is in every case no worse than the static and random policies and in most cases performs close to Optimal. It is important to recall that the time for performing the measurement probes is included in the OnePercent values presented in the graph. Thus, even with the measurement overhead accounted for, the OnePercent dynamic server selection policy is superior to static policies.

Turning to network load we find that in fact substantially less than 1 percent overhead was necessary to achieve these results. In fact, using the 500 byte (5 100 byte ping packets) overhead for the larger files results in an aggregate network bandwidth overhead of about 0.1 percent.

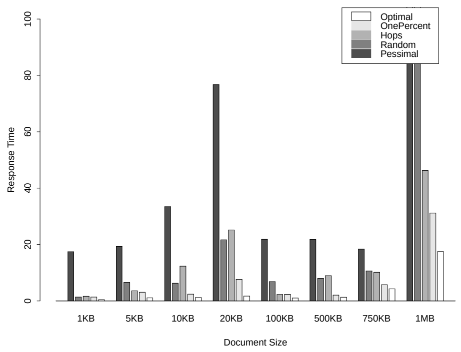

Figure 4: Performance of OnePercent dynamic server selection policy.

5 Future Directions and Conclusions

In order to deploy our dynamic server selection protocol in the current WWW environment we plan to modify a browser to support multiple URLs per link in a document. This list of alternate servers can be used as input to dynamic server selection by the browser. This represents a step along the way to integrated support for replication in which the browser would *obtain* a list of replicas.

We have also packaged the probe tools as a daemon that can be placed strategically throughout a network to allow probing of conditions in remote parts of the network. This will allow measurement of a link as a part of several paths and could be used to provide confirmation of probe estimates.

In conclusion, we have motivated the need for dynamic selection of servers by observing that the combination of random placement of replicas with dynamic server selection can result in a reduced average response time for service clients when compared with static policies. We presented simulation results that show the distinct benefit of dynamic server selection over standard static server assignment policies. We then illustrated how available bandwidth measurements (such as made by CPROBE) could be used by clients of replicated services as input to a server selection decision in the context of WWW documents, thus demonstrating the use of our tools in support of application-level congestion avoidance. In addition, we have also shown that most of the benefits of out detailed measurement approach can be achieved by a policy that adds less than 1% additional traffic to the network.

References

[1] Azer Bestavros. Demand-based resource allocation to reduce traffic and balance load in distributed information systems. In *Proceedings of SPDP'95: The 7^{th} IEEE Symposium on Parallel and Distributed Processing*, pages 338–345, San Anotonio, Texas, October 1995.

[2] Azer Bestavros, Robert L. Carter, Mark E. Crovella, Carlos R. Cunha, Abdelsalam Heddaya, and Sulaiman A. Mirdad. Application-Level Document Caching in the Internet. In *IEEE SDNE'95: The Second International Workshop on Services in Distributed and Networked Environments*, June 1995.

[3] Robert L. Carter and Mark E. Crovella. Dynamic Server Selection using Bandwidth Probing in Wide-Area Networks. Technical Report BU-CS-96-007, Boston University, March 1996.

[4] Robert L. Carter and Mark E. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. In *Performance Evaluation (Proceedings of Performance '96)*, volume 27&28, pages 297–318, Lausanne, Switzerland., 1996.

[5] Anawat Chankhunthod, Peter B. Danzig, Chuck Neerdales, Michael F. Schwartz, and Kurt J. Worrell. A hierarchical internet object cache. Technical Report CU-CS-766-95, University of Colorado - Boulder, Mar 1995.

[6] Yasuhiro Endo, James Gwertzman, Margo Seltzer, Christopher Small, Keith A. Smith, and Diane Tang. VINO: The 1994 fall harvest. Technical Report TR-34-94, Harvard University Department of Computer Science, 1994.

[7] James D. Guyton and Michael F. Schwartz. Locating nearby copies of replicated internet servers. In *Proceedings of SIGCOMM '95*, pages 288–298, Boston, MA, August 1995.

[8] James Gwertzman and Margo Seltzer. The case for geographical push-caching. In *HotOS '94*, 1994.

[9] net.Genesis Corporation. Comprehensive list of sites. Available at `http://www.netgen.com/cgi/comprehensive.`, April 1995.