Web Server Path Selection Using Lowest Latency to Client

Steve Padgett

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
spadgett@cc.gatech.edu

December 2, 2002

Abstract

A novel approach is presented to dynamically choose the lowest-latency path to a client from a web server. This approach works on a per-client basis and is based on inherent capabilities of the TCP and HTTP protocols and on the fact that high-availability datacenters contain multiple connections to the Internet. This method uses an established valid TCP connection to send duplicate packets, which in turn determine the link with the lowest latency. The costs of using this type of probing technique are explored, as well as ways to implement this different that would reduce the initial overhead in using this method.

1 Introduction

This paper focuses on the path selection to a client from a datacenter. Here, this method describes a way to dynamically choose the link with the lowest latency to the client so that the client's perceived wait time is reduced.

The method described below is a novel method, and is not described in any related papers to date. Unlike many related ideas that also strive to reduce latency to the client, it does not require any modifications to the server or to the client (no operating system changes, no extra applications) and is based simply off of the inherent properties of the TCP protocol.

Many high-availability datacenters have multiple Internet feeds to multiple providers. This way redundancy can be maximized and the datacenter can be positioned as close (in "Internet distance") to as many clients as possible. Previous studies have developed

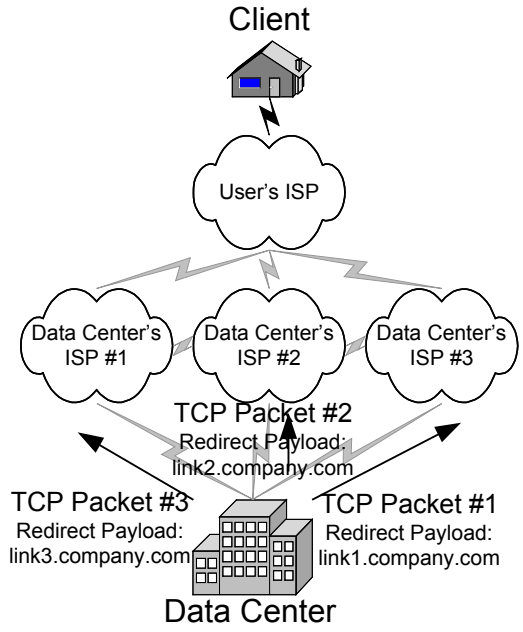

Figure 1. Duplicate TCP packets sent over the existing connection determine which link from the data center to the client provides the lowest latency. The packets have different payloads but the same TCP sequence numbers.

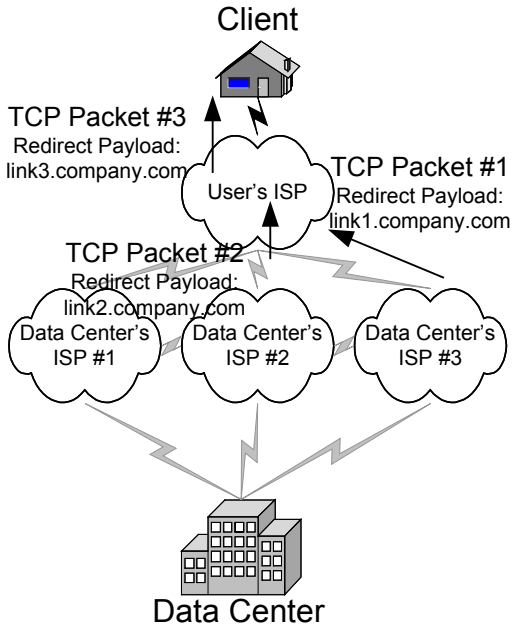

Figure 2. Packet #3 happens to get back to the client first, which follows the redirect to link3.company.com (which would be directed over the link that packet came from). The client discards TCP packets #1 and #2 as duplicate packets.

ways of directing clients to the closest possible datacenter [2,4,9,11,12]. Closest is typically defined as the datacenter that would provide the lowest possible latency to the client. However, because a single datacenter now typically connected to the Internet in multiple logical locations (via multiple providers), that datacenter has the ability to dynamically decide which link to use (which "logical" Internet location) to send the traffic from.

This paper is concerned with traffic leaving a single datacenter. A more global method of sending traffic to the fastest datacenter can still be used independent of the path selection out of the datacenter.

The link with the lowest latency to the user is dynamically selected at the initiation of the traffic flow. The purpose of this study is to provide a better user experience (as it is in [7] and [8]) by reducing the latency to the client. Similar devices, such as load balancers, content switches, and web switches strive to reduce latency on the web server end, by selecting the fastest possible web server. The method here of link determination would work in conjunction with standard load balancing equipment, both local (directing requests within a cluster of local machines in the datacenter) and global (based on DNS or some other method to direct requests to multiple datacenters) [2].

To implement this, a specialized machine would be deployed in the datacenter that would handle the initial incoming web traffic (because the initial DNS name that clients used to connect, such as www.company.com, would point to this machine). The link determination (LD) machine accepts the initial connection, dynamically has the client determine which Internet link coming out of the datacenter provides the lowest latency (again, without any software or configuration on the client), then redirects the client connect to the web servers in the datacenter that would utilize this chosen link. The client continues to use this link for the duration of the flow.

This method also does not require extra web servers (a set for each link) is not required. Nevertheless, an IP address for each website per Internet link (a VIP for each link) is needed so that the router can differentiate between traffic destinations. Policy-based routing is needed on the routers so that traffic sourced from each VIP would be sent out the designated link.

Finally, it is important to stress that changes are not required on either the client or the server end. An additional machine (two, for high-availability) would be needed to handle the initial connections. The client dynamically chooses the datacenter's Internet connection that provides it the lowest latency, and is redirected to the IP address (or DNS name) of the servers in the datacenter utilizing that particular connection.

2 Related Work

In [4], dynamic server selection based on RTT measurement is shown to significantly outperform static server selection. Also shown is that although 4 RTT measurements provides an optimal path a higher percentage of a time than other methods, anywhere from 1 to 4 measurements are still better than random or static server selection. The approach explained below uses one measurement done at the start of the http flow. Because of the work in [4], this measurement should provide a better path than simply basing the routing off simply the AS hop count or actual router hops.

Cardellini et al [2] discusses various approaches to request routing, such as triangulation and http redirection. DNS routing is also mentioned as providing server selection, however it is regarded as having extremely coarse-grained control over the process. The approach used in this paper uses http redirection and has the advantage of a more fine-grained approach than DNS routing.

Application layer anycasting [12] uses distributed servers and anycast groups to direct clients to the server with the lowest expected latency. This requires support on the client for the anycast resolver, and uses remote server probers to monitor network path performance. The advantage of the approach specified below is that it does not require any changes to be made to the client or to the server, as it monitors network path performance dynamically per host when that host connects to the datacenter.

Projects such as SPAND [9,11] are effective by relying on passive detectors to gather traffic patterns to allow servers to redirect clients to the fastest server possible. This method is shown to be highly successful; however, it is based off of deployment of new applications (or sniffers in the network) where the clients are located. Also, centralized database servers are required to store and report on the constant state of the network. In the method detailed in this paper below, the focus is on an individual datacenter, not a combination of traffic from multiple datacenters. Additionally, unlike SPAND, no changes are needed with the client or the server applications or system to implement the lowest latency path selection.

IDMaps [5] identifies latency as both being easy to measure and providing good measurements with just a few packets. Also, a possible correlation is indicated (although not explored) between latency and bandwidth – "applications that need high bandwidth [should] minimize latency as a first approximation." The correlations are explored in [1], where regression testing indicated a correlation between reduced latency and increased bandwidth.

Asymmetrical routing on the Internet it cited in [1] as the reason why packet round-trip-time is not a good source of latency measurement. That paper used ICMP timestamp requests to determine actual values for latency in one direction at a time. As explained below, actually measuring the values to produce a number for the latency (say, in terms of milliseconds) is not as important as the actual comparisons between the latency in the different paths, as that is what would affect the users' perception in loading the webpage.

Habib et al [6] show that 30% to 50% of the time in downloading a 1k file via HTTP is caused by network delay. The other delay is attributable to delay from DNS lookups and from the web server in parsing the request and finding the page. This shows that the

network-related delay is significant enough in the connection time whereas it should be improved if possible.

Other approaches have also been proposed for reducing latency on the web. TCP Fast Start [8] is a method for caching TCP timers between connections in order to avoid slow start. Persistent HTTP (P-HTTP) is discussed by Heidemann in [7] as a method to reduce the time to load web pages from a single server. Heidelmann shows P-HTTP to reduce load time by about 40% over standard HTTP.

3 Load Balancers

For high-volume high-availability Internet data centers, redundancy is implemented in the servers, in the switches and routers, and in the links coming into the datacenter. Web content can be replicated or distributed among servers, servers can have multiple redundant uplinks into switches, and routers can be physically redundant and have multiple redundant paths into the Internet. These paths may be through the same provider but different peering points (such as the datacenter peering with the same provider in both Atlanta and in Miami), multiple providers, or a combination of the two.

Dispatchers (also known as load balancers, content switches, or web switches) sit in front of the web servers and typically intercept the connections going to them [2]. They then decide, based their configuration and the state of the web servers, which server to route the connection to. Typically, dispatchers will monitor the health of the server and maintain a list of currently active servers so that it will only route traffic to servers that are within the established health guidelines. Because datacenters strive to maintain high availability, there would be multiple web servers capable of handling the same request. In order to differentiate between the web servers, the dispatcher may send the connections to the server based off of the server's load, the current number of connections to the server, the latency on the server, or any of several other variables. This way, clients will be routed to what would hopefully be the "best" server to handle their connection.

Nevertheless, even though the client may be routed to the web server with the lowest latency, the client may not be routed over the link out of the datacenter with the lowest latency across the Internet. Consider the case where the client is connecting over the same Internet provider that has multiple circuits into the datacenter, and these multiple circuits come from different locations (such as Atlanta & Miami). If the customer is in Miami, the gateway routers in the datacenter will not know (without special modifications, because the AS number is the same) that the provider's Miami link is preferable to the Atlanta link. Likewise, the packets sent via a particular provider may have to transit a saturated link, where the packets experience high queuing delay. In this case, routing the packets via an alternate link would be more appropriate, even if the packets were going to the original provider (assuming the packets don't have to transit similarly saturated links).

4 Lowest Latency Path Selection

The method this paper proposes for determining path selection is based off relatively simple concepts in TCP and HTTP. This method finds the path with the lowest latency from the server to the client. Latency from the client to the server is not measured because in a normal HTTP transaction, the client sends minimal amounts of data to the server (typically GET requests), while the server sends back larger amounts of data. Nevertheless, this latency is not harmed by this method either, as it's simply ignored. The TCP acknowledgements sent from the client to the server, though vital to the connection, are not under as significant of a time constraint to arrive back to the server.

Additionally, routing on the Internet can be asymmetrical. Packets traveling from the client to the server are not expected to take the same path as packets traveling from the server to the client. An attempt to measure the latency by measuring the round-trip-time of a packet (such as a with a ICMP echo/echo reply) would be misguided, since the single round-trip-time cannot be broken up into the request and the reply component parts [1].

In methods with similar goals, such as DNS-based server selection [10], anycasting [12], SPAND [9,11], and bprobe and cprobe [3], measurements are taken and recorded. Here, no actual measurement is taken. The client does not know how much less delay one link provides over the other, or the rankings of any of the links other than the fastest. There is no reason for the client to know this (as it simply needs to choose the link with the least amount of delay). This provides less administrative overhead in maintaining this method, because the data is dynamically generated when needed and never stored.

An attractive feature of this method is that it is supported by both clients and servers running any operating system that support the TCP protocol. Because this method is based off duplicate packet reception, and duplicate packets are allowed in TCP/IP, there should be no problem with compatibility in implementing this in a network. Additionally, because these packets are part of an already-established TCP connection, firewalls that would block ICMP probes from entering would not be able to block this method of path selection – the packets are all valid TCP packets for a flow already accepted by the firewall. Although there is a chance the firewall could filter out the duplicate packets (starting at the second packet it receives), the first packet (with the lowest latency) would have made it through, thus making the probe successful.

This method will work with or without distributors in front of the servers. In a high-availability environment, there are distributors in front of the web servers to monitor the server and route the traffic to an appropriate pool of servers. Using the link load balancing does not displace or remove the distributors from the network.

4.1 Latency

Link delay is generated from four different factors: transit, serialization, processing, and queuing delay. All of these contribute to the latency the user experiences when he accesses the Internet. According to Carter in [4], latency for a path can be estimated by as little as just one packet. Transit, serialization, and processing delay can be expected to remain reasonably consistent over a single path during the short period of time while a user is accessing a web site. Queuing delay only occurs when a link is congested and may not remain consistent. However, because queuing delay indicates that a link is congested, that link should typically be avoided in favor of a less congested link (if available). Thus, when a packet is slowed by delay from queuing on a link, that link and thus that path should be avoided if possible. This would naturally occur because the probe packet sent over the link that experienced the queuing delay would arrive after the other packets, and would not be chosen.

4.2 Methodology

Adding path selection / link determination to a network requires the addition of an additional server appliance, referred to here as the LD server. (Two LD servers could be added for a high-availability function).

The method for lowest latency path selection works as follows:

1. A client initiates a normal the TCP connection to the web server via TCP port 80. In this case, the DNS name for the server points to the LD server, so that the connection is actually established from the client to the LD.
2. A client issues a HTTP GET request to the LD server.
3. The LD sends back multiple HTTP redirects, one packet per link, each redirect containing *a different URL* for the client to connect to. The LD then closes the connection.
4. The client receives these packets, accepting the first one it receives and discarding the later packets as duplicate packets.
5. The client then establishes a new connection to the server it was redirected to, automatically choosing the link that has the highest available bandwidth.

The unique traffic flow actually occurs in step #3. The path selection occurs because the packets containing the HTTP redirect are not part of the normal TCP packet flow. Each packet contains a different payload (because the HTTP redirect address is different in each packet); however, each packet contains the exact same TCP sequence/acknowledgement numbers. In fact, the entire header (except for the checksums) is the same between all 4 packets. This causes the client, when it receives the packets, to think that the first packet it receives is the "good" packet. It copies the data from this packet into its buffers for the application (the web browser) to read. The client then receives the other packets that were sent with the same TCP sequence number. Because the client has already seen the data with that sequence number (with the same packet size), it considers the newly arrived packets to be duplicates and discards them. The client will then follow the redirect to the URL it received in the first packet. It connects to the server located on the link it received the first packet from, which is the link that experienced the smallest amount of delay to the client.

Each duplicate packet that is sent out (containing the same seq/ack numbers) contains a different HTTP redirect string. These

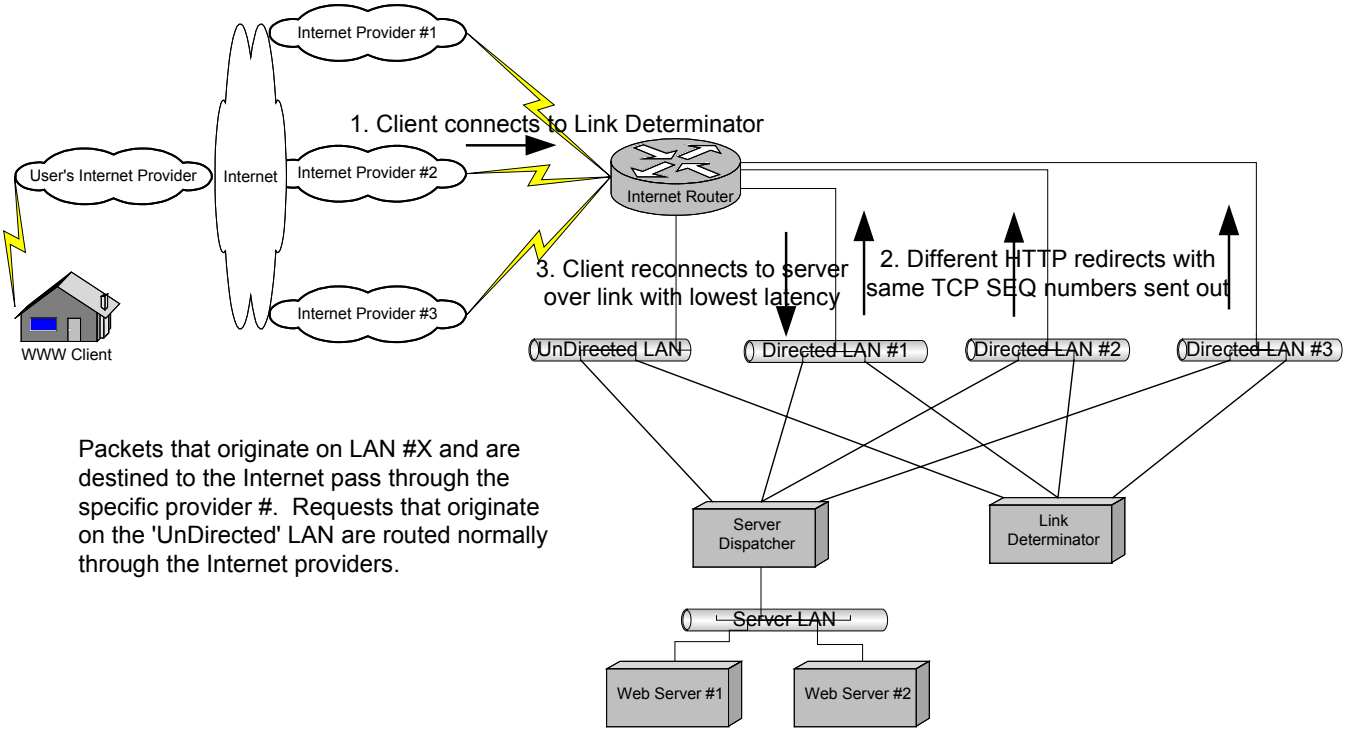

Figure 3: Link Determinator Setup

redirect strings are specifically designed so that the new connection made to these URLs will exit out of the same path as did the packet containing the redirect. This can be accomplished by assigning the web servers an IP address for each link then using policy-based routing to send packets sourced from each particular IP address out over a specific link.

In this way, the client automatically "selects" the link with the lowest latency and provides feedback to the server (via following the HTTP redirect) so that it receives future traffic over that link.

4.3 Implementation

For the purposes of experimentation, the LD was implemented using a modified Linux 2.4.19 kernel. The only modifications that were made to the kernel was the addition of extra fields to the getsockopt(. . . TCP_INFO . . .) call, to return the TCP sequence, acknowledgement, window sizes, and mss information about the connection. This way, a simple user-level call can be made with the socket as an option so that the application can receive the necessary TCP information about the socket. The packets are constructed in user-mode sent out over the wire using raw sockets.

The LD accepts the initial TCP connection normally, using the standard kernel interface (which provides the standard 3-way handshake). The application reads in the HTTP request through the standard system calls. When the HTTP redirect is ready to be sent, the LD software makes the appropriate kernel calls to gather the IP and TCP information about the connection, then formats and sends the specially created TCP packets using the raw Ethernet packet socket. Once these packets are sent, the client's TCP seq/ack numbers are out of sync with what the LD's kernel believes they should be. To remedy this, the LD software then sends the same TCP data using the connected socket that it had originally accepted the connection on, so that the kernel can synchronize its sequence numbers with what the client believes the numbers should be.

The packets are padded to the client's requested MSS in order to maximize the serialization delay of the link and to ensure that all packets are the same size. This way, the serialization delay, which contributes the most to bandwidth estimation, would be a more significant factor in the overall delay of the link.

The LD was set up in Netbed/Emulab and tested. Four wan emulators were connected to it in parallel, each providing a different delay. A client machine in front of the wan emulators connected to the LD via http and tested to see which path was chosen. The client machine then downloaded a 100k webpage from a web server sitting on the same networks as the LD. This page was downloaded simultaneously over all of the links in parallel to see which link actually provided the best results.

The load balancers were set up using the FreeBSD wan emulation software called dummynet. Delay was varied between trials, while bandwidth was held constant at 1MB/s. The tests were randomized so that that the lowest and highest delay values were randomly assigned among the 4 physical links. (This way the order of the physical links and the wan emulators would not influence the results). The links were tested with the delay varying between links by 1ms 2ms, and 3ms. (For instance, one of the trials in the 1ms category had delays of 35ms, 32ms, 34ms, and 33ms assigned to the links). A total of 1500 samples were taken (varying the link latencies each time) for each ms difference.

Table 1: Experiment Results

	1ms difference	2ms difference	3ms difference
Client chose best link	84.9%	96.0%	99.7%
Client chose 2nd best link	13.3%	4.0%	0.2%
Client chose 3rd best link	1.6%	0.0%	0.1%
Client chose 4th best link	0.1%	0.0%	0.0%

Table 1 shows the results for the experimentation. Even with the 1ms delay (which is the smallest delay increment that dummynet could provide), Table 1 shows that the best or second best link (which had 1ms latency more than the first) was selected 98.3% of the time. These results show that even a very small (1ms) of difference in latency between links is detected by this method with a high success rate.

5 Cost Considerations

This method uses active probe packets to perform the link determination. SPAND [9], which uses passive probe techniques, emphasizes that active probing can significantly reduce available network bandwidth. Though this is true, the amount of excess traffic sent is not nearly as large as other probers such as bprobe, cprobe, pathchar, packet bunch modes, or Treno [9]. For datacenters that have two Internet links, one extra packet would be sent per user, at the MSS size (a number negotiated by the client at the start of the connection). Dialup Internet users, the ones who typically the least available bandwidth, typically have a MSS setting approximately one-third that of higher speed connections; thus, dialup Internet users in this scenario would only experience about 600 excess bytes at the start of the visit to the website. A datacenter with four Internet links would generate 3 extra packets, which would still be less than 1.8kb of excess data for a dialup user (4.5kb for users with the MSS set to Ethernet frame sizes).

SPAND [9] claims that a correlation between latency and bandwidth does not exist. However, experimentation performed by Carter [4] contradicts this claim. Carter shows that RTT is correlated to available bandwidth and just a one-ping measurement at the initiation of the connection can provide up to a 35%-75% (based on the amount of data transferred) improvement in download time for the client. These results, corroborated in [5], show that using the method described here would provide the user with lower latency and a faster download time (assuming the user's connection isn't the bottleneck) than without using this method.

5.1 Performance Enhancement

As presented, this method requires two different TCP connections to be established. Both the initial connection to the link determination machine, then the new connection after the HTTP redirect to a separate machine are established. This requires two different TCP 3-way handshakes to be completed, which would add further latency to the connection. However, this could be implemented inside of a local load balancer. The connection would be made to the load balancer; the load balancer would send the TCP probes containing a redirect back to the same machine (same hostname, different path). The client, using standard HTTP/1.1 with persistent connections, would not need to close the connection and initiate a new one. The dispatcher, though, would send data under that virtual directory structure (that it redirected the client to) back out through a specific link to the router (this could be separate physical links or a separate logical links using VLAN tagging). The router would identify the link it came on and, using policy-based routing, send packets from that particular link (or VLAN) out the statically assigned wan link. Using this method, an additional TCP connection is not required (saving that overhead cost), and additional equipment would not be required for the datacenter, minimizing its cost. This could simply be rolled out in a software upgrade to the datacenter's existing dispatchers.

Additionally, work performed by Habib in [6] reports that the DNS lookup time takes 10%-25% of the time in downloading a 1k page from a web server. By locating this redirect in the dispatcher so that the client does not need to reconnect saves an extra DNS lookup, saving the client even further time over the initial method. Alternatively, the client could be redirected to an IP address instead of a DNS name saving this time as well. However, the client web browser's Address Bar will then typically show a IP address for the site instead of a name which may not be appropriate.

Finally, in using this revised method, an IP address for the web sites are not needed for each Internet link out of the datacenter. Previously, a VIP for each link was required, so that when the client connected to the VIP, the router would base its routing decision on the source IP of the packet (using policy-based routing). Now, the load balancer simply needs to route send the traffic back over a specific link (physical or logical, such as a tagged VLAN). The router then would need to base its policy routing decision on the incoming interface of the packet and not the IP address.

6 Other Applications

Although this paper focused on using this method with HTTP, this method of link determination could easily be extended for other uses. HTTP provides a built-in redirect function that allows the client to effectively provide feedback to the server about which packet it received first (signifying which link had the lowest latency). Other applications that strive for low latency could also use this method. For instance, an Internet gaming site with multiple Internet links could use a method similar to this to choose the fastest link its clients. A specialized multimedia-streaming solution could use this method as well to provide low latency in the multimedia stream.

7 Conclusion

This paper presented a new method of using the properties of TCP to find the lowest-latency path to a client. No changes are needed on either the web server or to the client to support this, as this would be implemented inside the datacenter as a separate machine. By sending out duplicate TCP packets that contain different payloads, the client is redirected to the link with the lowest latency. The client is then able to access the website over the path which it experiences the least delay out of the datacenter.

Additionally, by implementing this as part of an existing dispatcher as described in "Cost Considerations", a new connection would not be required for the HTTP redirect, saving an additional three-way TCP handshake.

This method has been tested experimentally. The results show that this method of path determination works properly in locating the link with the lowest latency. The client is able to dynamically choose the fastest link out of the datacenter.

8 References

[1] K. G. Anagnostakis and M. B. Greenwald. Direct measurement versus indirect inference for determining network-internal delays, March 2002. http://citeseer.nj.nec.com/anagnostakis02direct.html

[2] V. Cardellini, M. Colajanni, P. Yu. The State of the Art in Locally Distributed Web-server Systems. 2001. http://citeseer.nj.nec.com/cardellini01state.html

[3] R. Carter and M. Crovella, "Measuring Bottleneck Link Speed in Packet-Switched Networks," Performance Evaluation, Vol. 27-8, pp. 297-318, Oct. 1996. http://citeseer.nj.nec.com/carter96measuring.html

[4] R. Carter and M. Crovella. Server selection using dynamic path characterization in Wide-Area Networks. *IEEE Infocom'97*, 1997. http://citeseer.nj.nec.com/carter97server.html

[5] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "IDMaps: a global Internet host distance estimation service," IEEE/ACM Transactions on Networking, vol. 9, pp. 525–540, 2001. http://citeseer.nj.nec.com/francis00idmaps.html

[6] Md Ahsan Habib, Marc Abrams "Analysis of Sources of Latency in Downloading Web Pages", WebNet 2000, October 30 - November 4, 2000 San Antonio, Texas, USA http://citeseer.nj.nec.com/489740.html

[7] J. Heidemann. Performance interactions between P-HTTP and TCP implementations. ACM Computer Communication Review, 27(2):65--73, April 1997. http://citeseer.nj.nec.com/heidemann97performance.html

[8] V. Padmanabhan and R. H. Katz, "TCP fast start: a technique for speeding up web transfers," accepted for

publication for Globecom 1998 Internet Mini-Conference. http://citeseer.nj.nec.com/padmanabhan98tcp.html

[9] Srinivasan Seshan, Mark Stemm, and Randy Katz. Spand: Shared passive network performance discovery. In Proceedings of the USENIX Symposium on Internet Technologies and Systems, 1997. http://citeseer.nj.nec.com/article/seshan97spand.html

[10] Anees Shaikh, Renu Tewari, and Mukesh Agrawal, "On the Effectiveness of DNS-based Server Selection," Proc. of IEEE INFOCOM '2001, 2001. http://citeseer.nj.nec.com/article/shaikh00effectiveness.html

[11] M. Stemm, R. Katz, and S. Seshan. A Network Measurement Architecture for Adaptive Applications. Proceedings of IEEE Infocom 2000, March 2000. http://citeseer.nj.nec.com/stemm00network.html

[12] E. Zegura, M. Ammar, Z. Fei, and S. Bhattacharjee. Application-layer anycasting: A server selection architecture and use in a replicated web service. ACM/IEEE Transactions on Networking, 8(4):455-466, 2000. http://citeseer.nj.nec.com/411865.htm