Faster Web through Client-side DNS Redirection

Utkarsh Goel and Mike P. Wittie

Department of Computer Science,
Montana State University, Bozeman MT 59715,
{utkarsh.goel, mwittie}@cs.montana.edu

Abstract. Modern websites use Content Delivery Networks (CDNs) to speed up delivery of static content. However, we show that DNS-based load-balancing of CDN servers fails to fully deliver on the speedup of CDNs. We propose DNS-Proxy (`dp`), a client process that divides CDN load-balancing functionality between clients and CDNs by choosing from among resolved CDN servers based on last-mile network performance. Our measurement study shows that `dp` reduces webpage load time by 29%. If `dp` has already resolved the domain, the reduction in webpage load time is 37%. Finally, `dp` reduces the download time of individual static Web objects by as much as 60%. We argue that `dp` enables a more effective use of existing content delivery infrastructure and represents a complementary strategy to a continual increase of geographic content availability.

1 Introduction

The growing competition among Internet services such as online social networks, e-commerce, or streaming video, drives developers to improve responsiveness of their applications. Content Delivery Networks (CDNs) play a vital role in reducing request delay to improve the user experience [9]. To increase application responsiveness and retain users, businesses invest significant resources to geographically distribute content onto CDNs [6][17]. However, user frustration with Web content delivery indicates that CDN performance is far from expected [4].

The speed at which static content is delivered is dominated by network latency between clients and CDN servers [4][8]. Content-rich websites contain images large enough to require multiple round trips to download [22]. Further, website rendering includes dependencies, which means that static content such as image, advertisement, javascript, and CSS files are fetched over multiple request rounds [11]. CDNs reduce the impact of round-trip times (RTTs) on overall page load time by serving content from widely distributed servers in last-mile networks. CDNs balance load on their servers using DNS redirection, where geographically distributed DNS servers resolve CDN URLs to IP addresses of nearby content servers [1][2]. However, we discover that content distribution does not reduce the network latency as expected, because DNS load-balancing frequently does not direct clients to closest available content server.

Based on extensive measurement of CDNs used by Facebook and Google, we identify three shortcomings of the DNS load balancing mechanisms used by

CDNs. First, there is a high performance gap between CDN servers returned within a DNS response. Second, the CDN servers returned by one DNS server can perform remarkably better than the CDN servers returned by another DNS server. Although such differences are to be expected in general, we show they are domain dependent, in that the same DNS does not always provide the best CDN servers for a given client. And third, institutional networks are often configured with firewalls, which block access to some external servers and content hosted on them. Our study shows that DNS resolutions occasionally redirect client applications to inaccessible CDN servers, resulting in failed downloads of Web content even from well-known content providers such as Facebook and Google. We conclude that current application of DNS load-balancing fails to direct users to the best CDN server, which confines content delivery and application performance.

Based on our study of DNS redirections on Web content delivery, we argue that clients are best positioned in the network to measure CDN performance. It is therefore timely to explore solutions where server selection is shared by clients and CDNs to both minimize network delay and balance server load. We propose DNS-Proxy (`dp`), to make server selection aware of last-mile network performance. `dp` implements a lightweight parallel probing mechanism of resolved CDN servers to redirect client applications to the fastest one. Our results show that, `dp` reduces the webpage load time by 29%. If a request for a domain is already resolved by `dp`, the reduction in webpage load time is about 37%. `dp` reduces the download time of individual Web objects by 60%. To best of our knowledge, `dp` is the first step to extend CDN server selection to client devices. `dp` builds the foundation of a new paradigm where load balancing is a shared functionality between CDNs and client devices. We make `dp` available as an open source tool at `https://github.com/msu-netlab/DNS-Proxy`.

The rest of the paper is organized as follows. In Section 2 we outline the related work to reduce latency in Web communications. In Section 3, we illustrate the ineffectiveness of CDN/DNS systems. In Section 4, we discuss the implementation and benefits of `dp`. Finally, we conclude in Section 5.

2 Related Work

In spite of several years of effort to reduce client latency, Web applications still struggle to deliver a responsive experience [10]. Previous studies have proposed a number of techniques to minimize the impact of network latency on application performance. Specifically, some techniques reduce the Web latency through new communication protocols implemented in client browsers and the network stack, while others improve the accuracy of DNS redirections in directing applications to the fastest CDN server.

Browser techniques: Shared Dictionary Compression over HTTP (SDCH) is a communication protocol that relies on inter-response HTTP compression to reduce the amount of data transferred in the network and latency in data delivery [5]. Although SDCH enables faster access to commonly accessed Web objects, SDCH has not been widely adopted.

SPDY uses concurrent HTTP transfers on a multiplexed TCP connection to reduce the latency in Web communication [11]. SPDY can reduce the Web latency by 50%, but only if the Web server has more than six resources to offer to the client browser. Our approach, `dp`, offers complimentary benefits to SPDY by speeding up individual object downloads.

DNS techniques: EDNS mechanisms can CDNs to assign users to closest servers based on client subnet [19]. However, this approach is still limited by how quickly DNS caches can be updated.

Vulimiri *et al.* proposed client tools for sending DNS requests to multiple DNS servers and using the first received DNS response on the client [20]. CoDNS is a similar approach that distributes incoming DNS requests to other DNS resolvers to mask the delay in DNS lookups [14]. Although, these techniques reduce DNS lookup time, our study shows that the first DNS response may not redirect the user to the fastest available CDN server and may negate the effect of fast DNS resolutions.

Namehelp Mobile is a tool that measures the impact of cellular DNS redirections on mobile application performance and allows users to find DNS servers that may perform faster DNS lookups and faster Web downloads [15]. Although, Namehelp compares the performance of different DNS resolutions, it does not consider the performance of servers within a DNS response. Similar to Namehelp, Direct Resolution characterizes the impact of DNS redirections from remote DNS servers on CDN performance [12].

Another approach that aims to reduce DNS lookup time relies on reducing the DNS cache miss rate by exploiting similarity of request domains to domains already cached [18]. While this technique may speed up the look up of domain names, it may not speed up applications by redirecting clients to faster servers, since the technique does not consider CDN performance before redirecting client applications.

Other techniques: MONET improves the availability of websites to clients [3]. MONET uses link multi-homing in combination with cooperative overlay network of peer proxies to discover possible network paths between Web servers and clients. WhyHigh helps network administrators to diagnose the cause of latency inflation in delivering static content from CDN servers [10]. WhyHigh measures client latencies from all nodes in a CDN and identifies relationship between measurement and client prefixes affected by inflated latencies.

3 Load-Balancing by CDN Infrastructure

Our measurement study of CDNs used by Facebook and Google over four months shows that CDN based load-balancing techniques frequently do not redirect clients to the closest, or to the least-loaded server. As a result DNS load-balancing diminishes the speed up of CDNs to content delivery and application responsiveness.

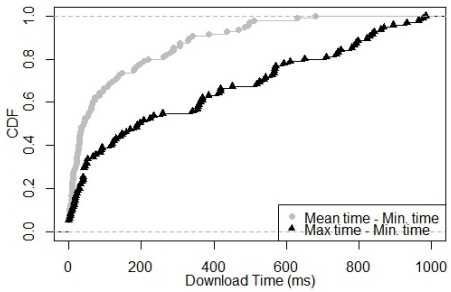

Fig. 1. Image download times from Facebook CDN servers.

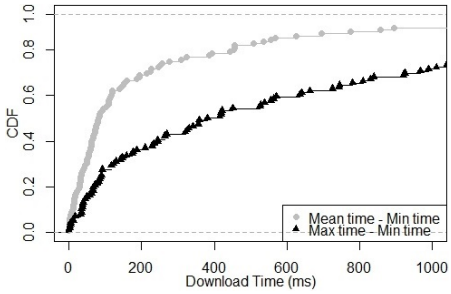

Fig. 2. Image download times from Google CDN servers.

3.1 Impact of CDN Choice on Content Download Time

To measure the variation of CDN server performance, we setup an experiment on 300 test devices obtained from Seattle [23] and Dasu [16] testbeds. We configured each device to send a DNS query to its local DNS, Google's public DNS, and two open DNS servers to resolve CDN addresses used by Facebook (to serve user photos) and Google (to serve advertisement images). Next, the test devices gathered DNS responses and initiated probes to measure the time to, one, establish a TCP connection, two, receive first bit of HTTP HEAD, and three, to download a static image, for each of the CDN servers in DNS responses.

We graph image download time from different CDN servers returned by client's local DNS in Figure 1 and Figure 2. Figure 1 shows a Cumulative Distribution Function (CDF) of 78 KB image download times from Facebook CDN servers. Figure 2 shows a CDF of 113 KB image download times from Google CDN servers. For consistency of display we cut off CDFs at 1000 ms. The black lines show the distribution of the difference of download times between the fastest and the slowest CDN server within a local DNS response. The gray lines show the distribution of the difference between the mean download time and the download time of the fastest CDN server. Operating systems choose the first server for a randomized list in a DNS response, and so the mean of download times respesents the expected performance of these random choices.

These graphs show that for most of the clients, the performance of the fastest CDN server in the DNS response is far from the performance of the slowest CDN server available and even from the mean download time. We suspect that this variation comes from the variability in time spent establishing TCP connections and internal server queueing. We verify these hypotheses next.

Figure 3 is a CDF of TCP connection setup time to Facebook CDN servers returned by each client's local DNS. The black lines show the distribution of the difference of TCP connection setup times between the fastest and the slowest CDN server. The gray lines show the distribution of the difference of TCP connection setup times between the mean and the slowest CDN server. Figure 3 shows that for most of the clients the TCP connection setup time to the slowest CDN server is more than that of the fastest CDN server. This difference can

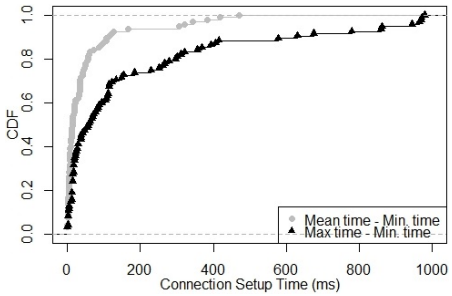

Fig. 3. TCP connection setup time to Facebook CDN servers.

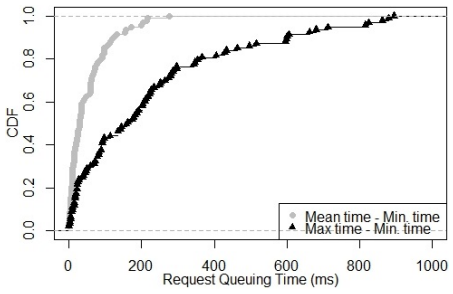

Fig. 4. HTTP request queuing time at Facebook CDN servers.

be as much as a second between slowest and fastest, and half a second between the mean and fastest server. This result indicates that clients might speed up content delivery if they select CDN servers based on TCP connection setup time.

Finally, request queuing on CDN server plays an additional role in influencing the performance of Web applications. High user traffic on a CDN server could result in high request queueing delays. The HTTP HEAD response time is a good indicator of the size of the request queue on a CDN server. Figure 4 shows a CDF of approximate HTTP request queuing (obtained by subtracting TCP connection setup time from HTTP HEAD response time) on CDN servers used by Facebook. We observe The black line shows the distribution of the difference of HTTP request queuing times between the least loaded and the most loaded CDN server, returned in the DNS response. The gray lines show the distribution of the difference between the mean and the most loaded server. We see that CDN servers returned in the DNS response have different request processing times and thus a poorly chosen server will increase the HTTP response time. We observed similar trends for the Google CDN servers, but omit the figures due to space constrains.

3.2 Impact of DNS Choice on Content Download Time

Analogous to the choice among CDN servers, clients also have a choice between DNS servers. Our study shows that a client misses an additional opportunity to reduce the Web latency when compelled to use a single DNS server, because the performance of CDN servers returned by one DNS server differs from the performance of CDN servers returned by another DNS server.

We extended our measurement study to evaluate the impact of DNS choice on Web performance. Specifically, we show the impact of DNS choice on image download time in Figure 5 for Facebook CDNs and in Figure 6 for Google CDNs. These figures show the difference in the image download times from the fastest server returned by local DNS and the fastest servers returned by Google's DNS and two Open DNS servers. We see from the figures that local DNS servers do not redirect clients to the fastest available CDN server at all times. Some clients and domain names may get faster results from Google DNS, while others from

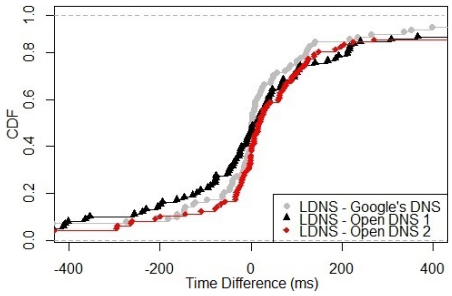

Fig. 5. Image download time from Facebook CDNs based on DNS choice.

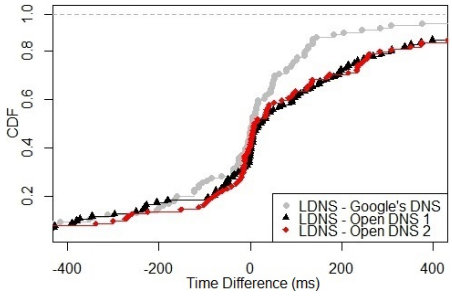

Fig. 6. Image download time from Google CDNs based on DNS choice.

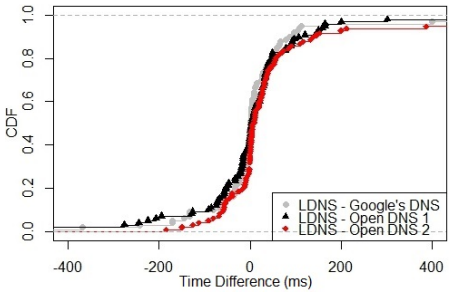

Fig. 7. Difference TCP connection times to Facebook CDNs based on DNS choice.

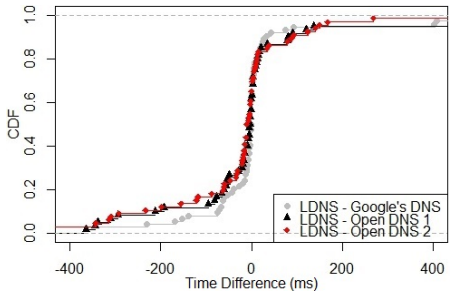

Fig. 8. Difference of request queue delay on Facebook CDNs based on DNS choice.

Open DNS. Therefore, it remains unclear which DNS server will redirect to the fastest server for a given domain and last-mile network. We depict similar trends from DNS choice in TCP connection setup time and HTTP request queuing time in Figure 7 and Figure 8, respectively. Similar to image download time, DNS redirections do not redirect clients to the server with lowest available TCP connection setup and HTTP request queuing time.

3.3 Overall Performance Variation

It is currently unclear how the choice of CDN and DNS impacts the Web performance when combined together. Therefore, we argue that it is timely to understand the combined impact of these choices on application performance and take necessary steps to continue growth in the Web space.

Figure 9 shows a CDF of the overall performance gap between the fastest, mean, and the slowest available CDN server obtained from multiple DNS servers. The black and gray lines show the distribution of image download times for CDNs used by Google and Facebook respectively. The circular point series depict the difference of the download time between the mean and the the fastest available server. The plus point series depict the difference of the download times between the slowest and the fastest available server. From the figure we see that for about 95% of the clients, the difference in the image download time between the slow-

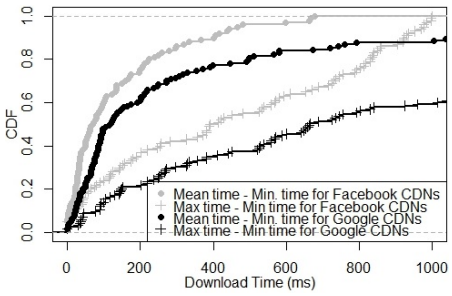

Fig. 9. Download time difference between CDN servers from multiple DNS servers.

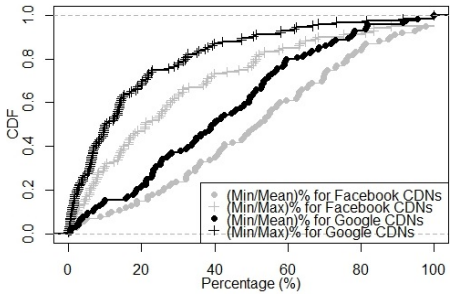

Fig. 10. Download time percent diff. between CDN servers from multiple DNSs.

est and the fastest available server is significant. Similarly, for about 80% of the clients the difference between the mean and the fastest server is significant.

In Figure 10, we depict the overall performance gap in percentage between CDNs used by Facebook and Google. The series with circular points show the ratio of the image download time of the fastest server to the mean image download time. The series with plus points show the ratio of the image download time of the fastest server to the image download time of the slowest server. We see that for the 85^{th} percentile of users connecting to Facebook CDN servers, the image download time for the fastest available server is only 60% of the slowest available server, resulting the image download time of the fastest server to be 40% faster than the slowest server. However, for Google CDN servers, the image download time for the fastest server is only 40% of the slowest available server, resulting the image download time of the fastest server to be 60% faster than the slowest server. Similarly for both Facebook and Google, the image download time for the fastest server is about 70% and 80% of that of the mean download time, respectively. This result indicates that there is a visible opportunity to reduce the Web latency even for well-established CDN services, which can be realized if client applications rely on multiple DNS servers for domain name lookups and evaluate the performance of each CDN server before using them.

3.4 Network Blockage to Reach CDN Servers

Client devices and end-networks are often configured with firewalls to block access to some IP addresses. Such restrictions in the network could be due to automated Intrusion Detection System (IDS) software that scan for any activity that might abuse network and device resources, which we verified in Montana State University's network. These network configurations are unknown to DNS servers and therefore when clients perform DNS lookups, they are occasionally redirected to inaccessible servers and result in failed downloads of Web content [21][13].

We recorded the percentage of faulty DNS lookups (that is, whether a DNS lookup contained an inaccessible server address) and the percentage of inaccessible CDN servers present within any DNS resolution. Our measurement study of over 19 thousand probes between different clients and CDN servers shows that

the probability of a DNS server returning a faulty resolution for both Facebook and Google is about 4% and the chances of an IP address being inaccessible in any DNS resolution are about 2%. Although server inaccessibility prevents the content to be downloaded all together, we believe that this could be prevented by vetting CDN servers. We therefore argue that new DNS based mechanisms are needed in the end-networks that will check server accessibility of CDN servers before redirecting client applications.

4 DNS-Proxy (`dp`)

Our measurements show that DNS load-balancing used by CDNs is not effective in meeting Web application requirements of fast and reliable content delivery. We propose DNS-Proxy (`dp`), a client process to select CDN servers based on access network performance. `dp` enables users to realize remarkable improvements in page load times and static content delivery in general.

`dp` operates as a virtual DNS server for client devices and adapts DNS resolutions according to current performance of end-networks. `dp` intercepts DNS requests from client devices and fans them out to multiple DNS servers and then probes resolved IP addresses before returning the fastest CDN to the client. `dp` can also be configured in DHCP to operate as a DNS server on network gateways to serve incoming DNS requests from multiple devices in the same network. We make `dp` available as an open source tool at `https://github.com/msu-netlab/DNS-Proxy`.

4.1 Network Aware DNS Redirections.

A sequence diagram representing `dp`'s mechanism to provide clients with the fastest server is depicted in Figure 11. `dp` listens for incoming DNS requests from clients on port 53. When a new DNS request arrives, `dp` forwards the client's DNS request to a number of DNS servers and waits for DNS responses. Upon receiving a DNS response, `dp` sends TCP SYN packets in parallel over raw sockets to port 80, (used by HTTP based services) and port 443 (used by HTTPS based services) of each server in the DNS response. The performance of each server in the DNS response is measured based on the time taken by `dp` to receive a TCP SYN/ACK packet from the server. To prevent `dp` from inadvertently launching a SYN attack, `dp` sends a FIN packet after receiving a SYN/ACK for each SYN packet, or after a timeout. Based on the performance measurement of each server, `dp` sends the fastest server back to the client in the DNS response. Although probing introduces extra load, it is important to not that `dp` caches probe results to avoid probing same servers in subsequent requests. Because `dp` reduces the number of requests leaving the network, by satisfying them from it's cache, the overall network load is reduced. In our experiments with a 24 hour DNS traces from within MSU's network we found that `dp` used less than 700 KB for every 500 domains even with very long timeouts.

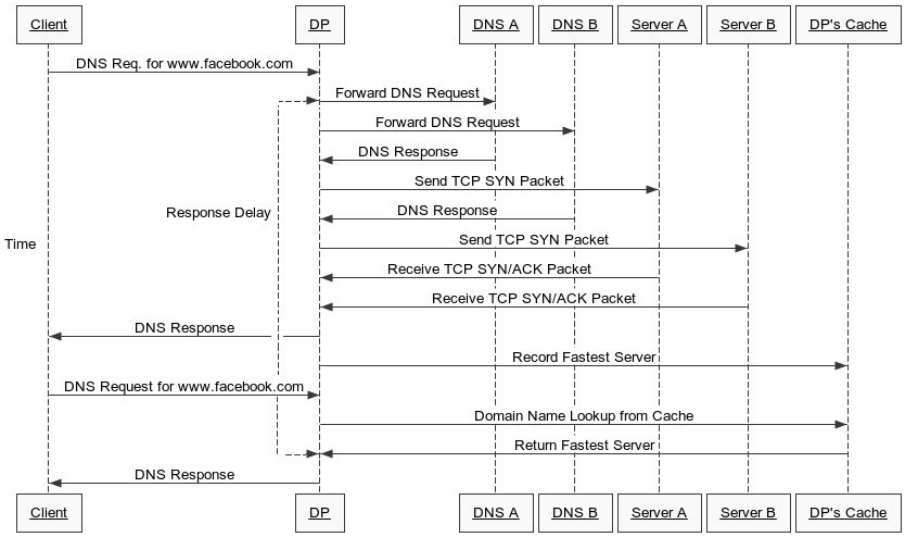

Fig. 11. Sequence diagram for DNS-Proxy functionality.

Our decision to use TCP connection setup time as an indicative of the CDN performance is based on our measurement data collected from Dasu test devices. We understand that predicting the fastest available CDN server based on Web content download time will reflect the actual Web performance, however, downloading Web content from each CDN server would introduce high probing traffic and take long time. Therefore, we argue that the prediction of fastest available server should be based on faster server selection mechanisms such as HTTP response time or TCP connection setup time.

To select the appropriate mechanism to predict the fastest server, we configured an experiment to probe resolved server IPs using TCP connection setup time and HTTP response time and select the fastest server using each approach. We allow `dp` different probing delays before returning the fastest found server to the client. The client then times the download of content from the returned server.

We show the download time from the fastest server identified after different probing delays in Figure 12 and Figure 13, for CDNs used by Facebook and Google respectively. In these figures, the x-axis marks different probing delay, while the y-axis marks the download time of the fastest server identified within the probing delay.

From the graphs, we see that as the probing time increases `dp` identifies a server which has lower download time. Moreover, we see that although the fastest server identified by either TCP connection setup time or HTTP response time results in identifying the server with same download time eventually, we show that TCP connection setup time is faster in that process. For example, in Figure 12, `dp` identified the fastest server, based on least TCP connection time, within 180ms, whereas, `dp` identified the server with the same download time, based on least HTTP response time, at 250ms. Therefore, with the knowledge

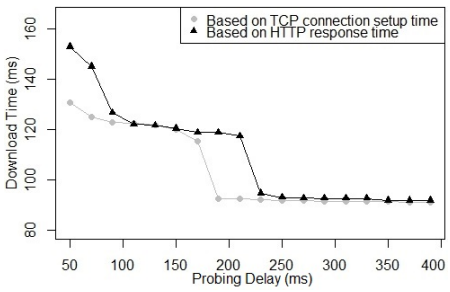

Fig. 12. Comparison of fastest server selection strategy for Facebook CDNs, based on TCP connection setup time, and HTTP response time.

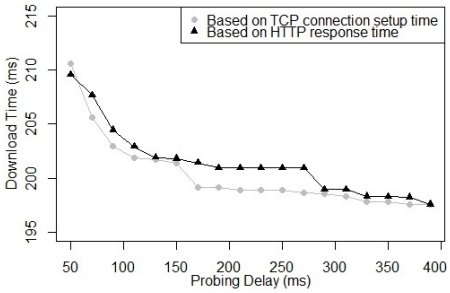

Fig. 13. Comparison of fastest server selection strategy for Google CDNs, based on TCP connection setup time, and HTTP response time.

that evaluating CDN performance based on TCP connection setup will require less time than evaluating CDN performance based on HTTP response time.

`dp` sends DNS responses to clients using one of two methods. `dp` responds either from its own cache of DNS entries (warm cache), or delays the DNS response for a domain not in the cache to probe for the fastest server on the fly (cold cache). Although, delaying a DNS response creates a perception of inflating the DNS lookup time and consequently the webpage load time, our study, however, shows that `dp` reduces the overall webpage load time when it redirects clients to faster CDN servers by delaying DNS responses. Describing Figure 12 again we show that delaying DNS responses will result in DNS redirections to faster servers. For example, delaying the DNS response by 20ms (from 170ms to 190ms), the image download time from the fastest server identified reduced from 115ms to 90ms. Therefore, a 20ms of delayed DNS response will result in speeding the image download time by 25ms and reduce the webpage load time even more, especially when multiple images are needed for the webpage from the same CDN server.

Finally, we show the reduction in webpage load time from using `dp` as a DNS server using 40 ms probing delay, for four major CDNs used by different websites in Figure 14. We show the corresponding reduction in webpage load times as percentage in Figure 15. We compare the webpage load time for websites, when clients in MSU's network use their local DNS and `dp` for DNS redirections. For this measurement, we picked websites that take advantage of some of the major CDNs and that have variation in the type of Web content they host. From the figures, we see that `dp` achieves different levels of speedups for different websites. For sites with many images, such as `huffingtonpost.com` and many adult tube sites, `dp` delivers a statistically significant speedup (37%) that is also meaningful to users (almost 3.5 s reduction of load time). Even for sites with relatively few images, such as `marvel.com` and `maps.google.com`, `dp` offers some speed and even on first request, when `dp`'s cache is cold. These results show that `dp` is effective in getting around inefficiencies of DNS load balancing to speed up page load times.

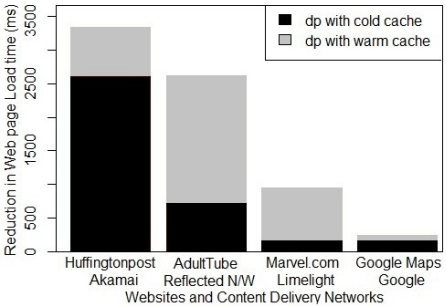

Fig. 14. `dp` speedup of webpage load time for four major CDNs.

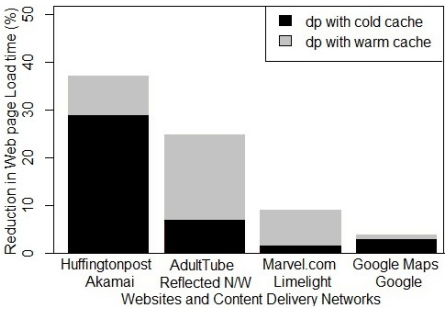

Fig. 15. `dp` speedup percent of webpage load time for four major CDNs.

4.2 Elimination of Inaccessible Servers from DNS Redirections.

Apart from redirecting client applications to the fastest available CDN server, `dp` probing eliminates locally inaccessible servers from DNS responses. However, if a DNS response contains only one IP address, `dp` redirects the user to that IP address, regardless of it being inaccessible. The process to check whether an IP address is accessible relies whether the TCP SYN/ACK packet was received by `dp` from the server.

5 Conclusions

Web application performance is affected by DNS redirections to slow CDN servers. Although, DNS redirections point clients to nearby CDN replicas [7], we show that DNS load balancing is ineffective and that clients can achieve faster downloads by choosing among CDN and DNS servers. We argue instead for clients to choose CDN servers base on last-mile network performance. DNS-Proxy (`dp`) probes resolved CDN addresses and replies to client with the fastest available server. Effectively `dp` shares load balancing functionality with CDNs by selecting from a set of servers returned by DNS. In our experiments, `dp` significantly reduces webpage and static object load time, even in cold-cache scenarios. Overall we believe `dp` enables a more effective use of existing CDN infrastructure and represents a complimentary strategy to a continual increase of geographic content availability and browser approaches, such as SPDY and SDCH.

References

1. Everything You Need To Know About CDN Load Balancing. `http://www.webtorials.com/main/resource/papers/Dyn/paper1/CDN-LoadBalancing.pdf`, Sept 2014.
2. Traffic Director. `http://dyn.com/traffic-director/`, Sept 2014.
3. D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. N. Rao. Improving Web Availability for Clients with MONET. In *USENIX NSDI*, May 2005.
4. M. Belshe. More Bandwidth Doesn't Matter (much). `https://docs.google.com/a/chromium.org/viewer?a=v&pid=sites&srcid=Y2hyb21pdW0ub3JnfGRldnxneDoxMzcyOWI1N2I4YzI3NzE2`, Apr 2010.

5. J. Butler, W. Lee, B. McQuade, and K. Mixter. A Proposal for Shared Dictionary Compression over HTTP. http://lists.w3.org/Archives/Public/ietf-http-wg/2008JulSep/att-0441/Shared_Dictionary_Compression_over_HTTP.pdf, Sept 2008.
6. B. Forrest. Bing and Google Agree: Slow Pages Lose Users. http://radar.oreilly.com/2009/06/bing-and-google-agree-slow-pag.html, Jun 2009.
7. M. J. Freedman, K. Lakshminarayanan, and D. Mazières. OASIS: Anycast for Any Service. In *USENIX NSDI*, May 2006.
8. I. Grigorik. Latency: The New Web Performance Bottleneck. https://www.igvita.com/2012/07/19/latency-the-new-web-performance-bottleneck/, Jul 2012.
9. T. Hopkins. What Are The Benefits Of Using a CDN? http://www.rackspace.com/knowledge_center/frequently-asked-question/what-are-the-benefits-of-using-a-cdn, Sept 2012.
10. R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving Beyond End-to-end Path Information to Optimize CDN Performance. In *ACM IMC*, Nov 2009.
11. M. Belshe and R. Peon. SPDY Protocol. http://tools.ietf.org/html/draft-mbelshe-httpbis-spdy-00, Feb 2012.
12. J. S. Otto, M. A. Sánchez, J. P. Rula, and F. E. Bustamante. Content Delivery and the Natural Evolution of DNS: Remote DNS Trends, Performance Issues and Alternative Solutions. In *ACM IMC*, Nov 2012.
13. V. N. Padmanabhan, S. Ramabhadran, S. Agarwal, and J. Padhye. A Study of End-to-end Web Access Failures. In *ACM CoNEXT*, Dec 2006.
14. K. Park, V. S. Pai, L. Peterson, and Z. Wang. CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups. In *USENIX OSDI*, Dec 2004.
15. J. P. Rula and F. E. Bustamante. NAMEHELP MOBILE. http://aqualab.cs.northwestern.edu/projects/237-namehelp-mobile, Aug 2014.
16. M. A. Sánchez, J. S. Otto, Z. S. Bischof, D. R. Choffnes, F. E. Bustamante, B. Krishnamurthy, and W. Willinger. Dasu: Pushing Experiments to the Internet's Edge. In *USENIX NSDI*, Apr 2013.
17. N. Shalom. Latency is Everywhere and it Costs You Sales - How to Crush it - My Take. http://natishalom.typepad.com/nati_shaloms_blog/2008/12/latency-is-everywhere-and-it-costs-you-sales-how-to-crush-it-my-personal-take-away.html, Dec 2008.
18. H. Shang and C. E. Wills. Piggybacking Related Domain Names to Improve DNS Performance. *Computer Network*, 50(11), Aug 2006.
19. S. Souders. Extension Mechanisms for DNS (EDNS(0)). http://tools.ietf.org/html/rfc6891, Apr 2013.
20. A. Vulimiri, P. B. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker. Low Latency via Redundancy. In *ACM CoNEXT*, Dec 2013.
21. J. Wilkinson. Some images not loading on facebook. https://www.facebook.com/help/community/question/?id=10203329172006739, 2014.
22. M. P. Wittie, V. Pejovic, L. Deek, K. C. Almeroth, and B. Y. Zhao. Exploiting Locality of Interest in Online Social Networks. In *ACM CoNEXT*, Nov 2010.
23. Y. Zhuang, A. Rafetseder, and J. Cappos. Experience with Seattle: A Community Platform for Research and Education. In *GENI Research and Educational Workshop (GREE)*, Mar. 2013.