Faster Web through Client-assisted CDN Server Selection

Utkarsh Goel, Mike P. Wittie, and Moritz Steiner[†]
Department of Computer Science, Montana State University, Bozeman MT USA 59717
[†]Akamai Technologies, Inc., San Francisco, CA, USA 94103
{utkarsh.goel, mwittie}@cs.montana.edu, moritz@akamai.com

***Abstract*—Modern websites use Content Delivery Networks (CDNs) to speed up the delivery of static content. However, we show that DNS-based selection of CDN servers can be refined to fully deliver on the speedup of CDNs. We propose DNS-Proxy (dp), a client-side process that shares load-balancing functionality with CDNs by choosing from among resolved CDN servers based on last mile network performance. Our measurement study of CDN infrastructure deployed by five major CDN providers shows that dp reduces webpage load time by 29% on average. If dp has already resolved the domain, the reduction in webpage load time is as much as 40%. Finally, dp reduces the load time of individual static Web objects by as much as 43%. 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.**

***Index Terms*—DNS resolution, server selection, measurement**

I. Introduction

The growing competition among Internet services such as online social networks, e-commerce, or streaming video, drives developers to improve the responsiveness of their applications. Content Delivery Networks (CDNs) play a vital role in reducing the request delay to improve the user experience [1]. To increase application responsiveness and attract content providers, CDNs invest significant resources to geographically distribute their content servers [2]. However, users' opinions on Web content delivery indicates that CDN performance could be improved to meet user expectations [3].

The speed at which static content is delivered is dominated by the network latency between clients and CDN servers from which the content is downloaded [4][5][6]. Content-rich websites contain images large enough to require multiple round trips to download [7]. Further, website rendering on browsers includes dependencies, which means that static content such as image, advertisement, JavaScript, and CSS files are fetched over multiple request rounds [6][8].

CDNs reduce the impact of network round-trip times (RTTs) on overall page load time by serving content from widely distributed servers in last-mile networks. CDNs balance the load on their servers through DNS-based server selection, where geographically distributed DNS servers resolve CDN URLs to IP addresses of nearby content servers [9][10]. However, we discover that content distribution may not reduce the network latency as expected, because DNS-based server selection does not always direct clients to the closest available content server, for various reasons such as content availability, load-balancing, cost of bandwidth, etc.

We performed an extensive measurement effort to evaluate the performance of CDN infrastructure deployed by Akamai Technologies and Google Inc. and discovered the following four limitations of the current DNS-based server selection mechanisms.

- The end-to-end latencies between a client and the CDN servers returned in a DNS resolution may have a high variation.
- The end-to-end latency between a client and a CDN server resolved by one DNS server could be remarkably lower than the end-to-end latency between the same client and a different CDN server resolved by another DNS server. Although such differences in end-to-end latency are to be expected in general, we show that they are domain dependent, in that the same DNS does not always provide the fastest CDN server for a given client for every resolved Web domain.
- A DNS server may direct a client to unnecessarily distant CDN servers when closer CDN servers are available. Again, although such direction may result from intended load-balancing, we show how their negative impact may be avoided.
- DNS-based server selection may occasionally direct clients to CDN servers inaccessible from clients' network, which results in the failure to access static Web content.

We conclude that current implementation of DNS-based server selection should be improved, in that it should take full advantage of geographic server availability to reduce the impact of network RTT on overall webpage load time.

Based on our measurement study we argue that clients are best-positioned in the network to measure CDN performance and participate in the selection of the best CDN replica. It is therefore timely to explore solutions, where server selection is shared by clients and CDNs to both minimize the network delay and balance server load. We propose DNS-Proxy (dp), that complements DNS-based server selection with client measurement from the last-mile networks. dp implements a lightweight parallel probing mechanism to probe resolved CDN servers, to direct clients to the fastest available CDN server.

Our results show that, dp reduces the webpage load time by 29% on average. If a request for a domain is already resolved by dp, the webpage load time is reduced by as much as 40%. Finally, dp reduces the load time of individual static Web objects by as much as 43%. To the best of our knowledge, dp is the first step in this direction that extends the CDN replica selection functionality to client devices in last-mile networks on a per-domain granularity.

Our experience with dp shows that load balancing can be shared effectively between CDNs and client devices in end-networks. We make dp available as an open source tool at http://github.com/msu-netlab/dp.

Although some CDNs may use anycast for global load-balancing [11], the goal of this study is to understand the impact of DNS-based server selection used by most major CDNs.

The rest of the paper is organized as follows. In Section II, we describe our experimental setup and discuss the impact of current DNS-based server selection techniques on Web performance. In Section III, we discuss the implementation of dp as a tool for client-assisted server selection. Section IV describes our evaluation results. In Section V we offer a discussion of dp's path to deployment. In Section VI we outline the related work on reducing network latency for Web applications. Finally, we conclude in Section VII.

II. DNS-based Load Balancing

To discover the limitations of current DNS-based server selection techniques and to understand their impact on Web performance, we configured several experiments on 123 devices, made available by the Dasu testbed, in different last-mile networks across different geographic areas [12]. Our measurement data contains 887 DNS resolutions from 386 DNS servers and 9,040 TCP and HTTP GET probes from clients on different continents to 1684 distinct CDN servers. We show the geographic distribution of our test devices in Table I.

A. Experimental Setup

We measured the difference in end-to-end latency and download time of static content between clients and CDN servers returned by different DNS servers. We configured experiments on each device to download images from each of the resolved CDN servers, and record the time to establish TCP connection, time to receive the first bit of the HTTP response, and the time to download the image. Each device was configured to download a 77 KB image hosted on Akamai's CDN and a 118 KB image hosted on Google's CDN.

We configured each device to send a DNS query to its default (local) DNS server (LDNS), the Google's public DNS server 8.8.8.8 (GDNS), an open DNS server 208.67.222.222 (ODNS), and Level3's public DNS server 209.244.0.3 (L3DNS) to resolve domain names hosted by Akamai Technologies (fbcdn-profile-a.akamaihd.net) and Google Inc. (lh3.googleusercontent.com).[1] Next, each device recorded the resolutions from DNS servers, initiated a TCP connection with each CDN server in the DNS resolution, and recorded the time for TCP connection establishment. After a successful TCP connection, the device sent an HTTP HEAD request to warm up the CDN's cache and issued another HTTP HEAD request to record the time to receive first bit of HTTP HEAD response. Finally, each device sent an HTTP GET request to download the cached image and recorded the time to completely download the image.[2]

[1]We ensured that the LDNS configured on the device was not one of the open DNS servers used in our study.

[2]We ensured that the devices issued all the DNS queries and content fetches from CDN servers within a very small time window.

CONTINENT	# CLIENT DEVICES
North America	54
Europe	35
Asia	14
Australia	10
Africa	6
South America	4

TABLE I
GEOGRAPHIC DISTRIBUTION OF DASU NODES.

B. Impact of CDN Choice on Static Content Delivery

The DNS resolutions contain a list of IP addresses of CDN servers within a single subnet. The client operating systems selects the first IP address from the list. A DNS resolution contains a list of IP addresses of CDN servers, which might suggest that latency to these servers should be similar and that selecting any of the CDN IP addresses would not impact the download time. However, we show that there is a significant difference in the end-to-end latency between the client and each of the CDN servers/clusters returned by DNS.

In Figures 1-4 we show the minimum, average, and maximum difference in the end-to-end latency (measured as time to establish a TCP connection) among Akamai CDN servers resolved by LDNS, GDNS, ODNS, and L3DNS. Similarly, in Figures 5-8, we show latency difference among Google CDN servers. These graphs show that the end-to-end latency between the client and the CDN servers returned in the list has a high variation. For example, in Figure 1 we see that if clients always pick the server with the least end-to-end latency, 80 percent of the clients will have an end-to-end latency within 50 ms. However, since clients' operating systems pick the first CDN IP address from the list of resolved CDN servers, which results in a random CDN selection over time, we see that 80 percent of the clients will have an end-to-end latency within 100 ms. Therefore, if clients choose the first CDN IP present in the list they may not connect with the fastest server available, since the server with the lowest end-to-end latency with the client might not be the first server in the list.

Content providers are interested in understanding the impact of server selection on static content download times from CDNs. Therefore, next we show the impact of CDN choice on the image download time. In Figures 9-12, we show the minimum, average, and maximum image download time from Akamai CDNs returned by LDNS, GDNS, ODNS, and L3DNS. Similarly, in Figures 13-16, we show the minimum, average, and maximum image download time from Google CDNs returned by LDNS, GDNS, ODNS, and L3DNS. Similarly to previous graphs representing the variation in end-to-end latency, we show a variation in the download time of individual static Web objects. For example, in Figure 9, the minimum image download time for 80% of the clients is within 250 ms, however, the average image download time is within 750 ms. The difference in minimum and average image download time is within 500 ms for 80% of clients, which is due to the multiple round trips between clients and CDN servers involved.

C. Impact of DNS Choice on Static Content Delivery

Analogous to the choice among CDN servers, clients also have a choice between DNS servers. Clients have a number of options from which to chose their default DNS servers.

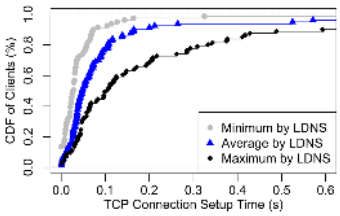

Fig. 1. Latency to Akamai servers resolved by LDNS.

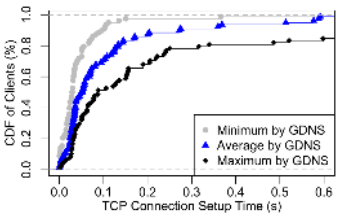

Fig. 2. Latency to Akamai servers resolved by GDNS.

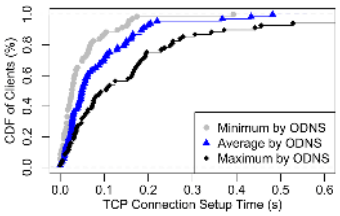

Fig. 3. Latency to Akamai servers resolved by ODNS.

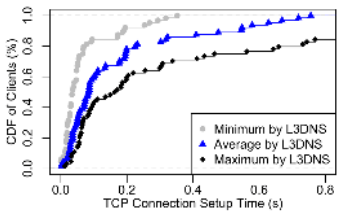

Fig. 4. Latency to Akamai servers resolved by L3DNS.

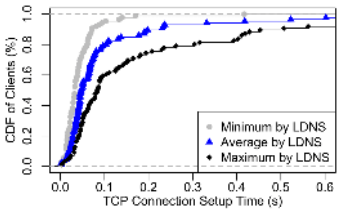

Fig. 5. Latency to Google servers resolved by LDNS.

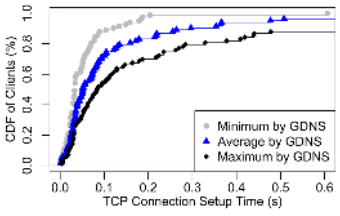

Fig. 6. Latency to Google servers resolved by GDNS.

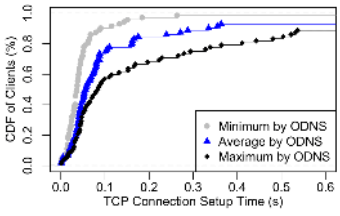

Fig. 7. Latency to Google servers resolved by ODNS.

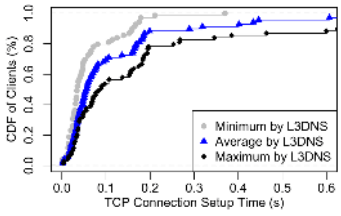

Fig. 8. Latency to Google servers resolved by L3DNS.

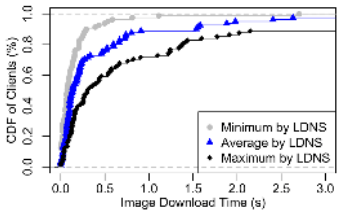

Fig. 9. Download time for Akamai servers resolved by LDNS.

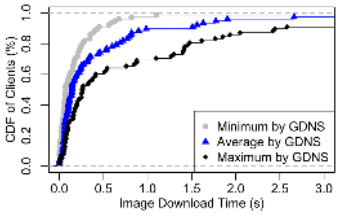

Fig. 10. Download time for Akamai servers resolved by GDNS.

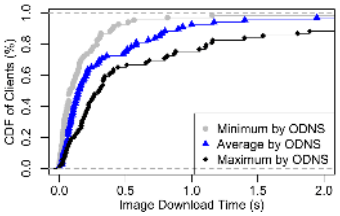

Fig. 11. Download time for Akamai servers resolved by ODNS.

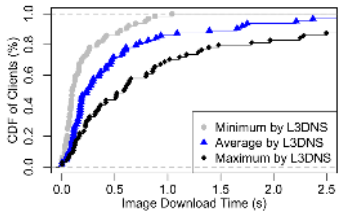

Fig. 12. Download time for Akamai servers resolved by L3DNS.

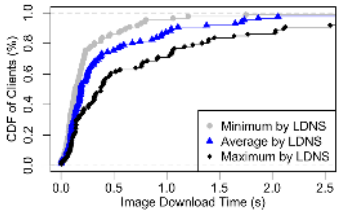

Fig. 13. Download time for Google servers resolved by LDNS.

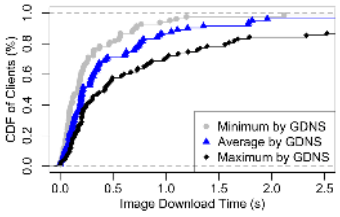

Fig. 14. Download time for Google servers resolved by GDNS.

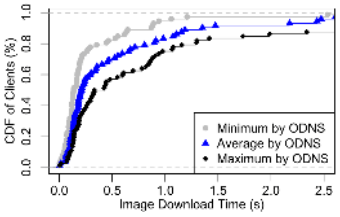

Fig. 15. Download time for Google servers resolved by ODNS.

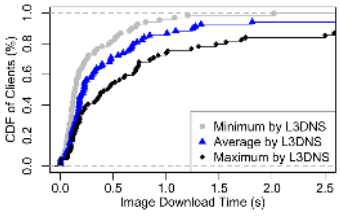

Fig. 16. Download time for Google servers resolved by L3DNS.

Further, some DNS providers may process EDNS-based DNS requests [13], while others may strip out any information available in the EDNS payload [14]. Thus, the variation in the adoption of EDNS mechanisms may also introduce additional variation in the performance of the CDN servers returned in a DNS resolution, since DNS servers that process EDNS-based requests could use the client's IP address available in the EDNS payload to direct the user to a CDN server nearest to the client's subnet. Finally, using a client's IP address from the EDNS payload may direct the client to the closest CDN server, however, the client may not have the least end-to-end latency with that CDN server due to congestion in the network, large queues on the server, or circuitous routing. Therefore, it is important to understand the performance of CDN servers returned by different DNS providers.

In Figures 17-20, we compare the end-to-end latency and the time to download images from Akamai CDN servers returned by LDNS, GDNS, ODNS, and L3DNS. Similarly, in Figures 21-24, we compare the end-to-end latency and the time to download images from Google CDN servers returned by LDNS, GDNS, ODNS, and L3DNS. These figures show that the end-to-end latency between clients and CDN servers returned by one DNS server is lower than CDN servers returned by another DNS server. For example, in Figure 18 we see that for 80% of the clients that resolve Akamai CDN domains from LDNS, GDNS, ODNS, L3DNS have an average end-to-end latency within 100 ms, 125 ms, 150 ms, and 300 ms respectively. Similarly, in Figure 22 we see that for 80% of the clients that resolve Google CDN domains from LDNS, GDNS, ODNS, L3DNS have an average end-to-end latency within 100 ms, 150 ms, 175 ms, and 200 ms respectively. Such variation in the end-to-end latency to CDN servers is also reflected in the time to download images from Akamai and Google CDNs. For example, in Figure 24 we see that for 80% of the clients the time to download image from LDNS, GDNS, ODNS, and L3DNS is 600 ms, 700 ms, 750 ms, and 800 ms respectively.

In Figure 18 we also see that for 35% of clients ODNS returns CDN servers with average end-to-end latency lower than CDN servers returned by GDNS. Similarly, for about 40% of the clients in Figure 22 we see that the average end-to-end latency to the CDN servers returned by all DNS servers are almost similar. As a result of such variation in DNS resolutions, it remains unclear which DNS server will direct the client to the fastest server at all times. We argue that clients could remain unaware of opportunities to reduce the Web latency when they rely on one DNS server.

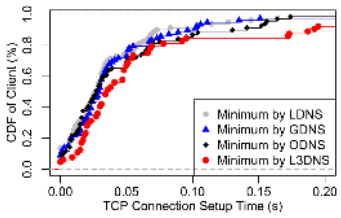

Fig. 17. Min. end-to-end latency from Akamai CDNs.

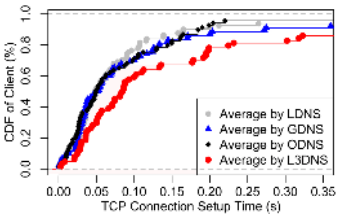

Fig. 18. Avg. end-to-end latency from Akamai CDNs

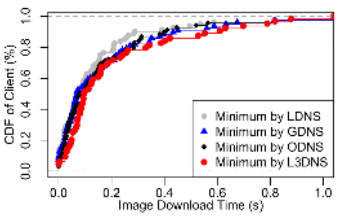

Fig. 19. Min. Image download time from Akamai CDNs

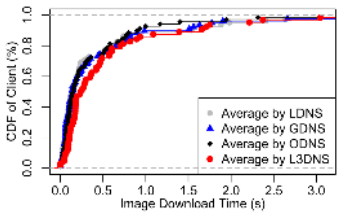

Fig. 20. Avg. Image download time from Akamai CDNs

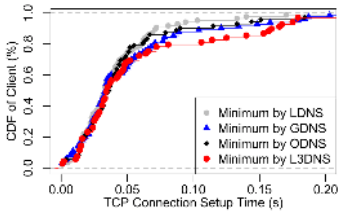

Fig. 21. Min. end-to-end latency for Google CDNs.

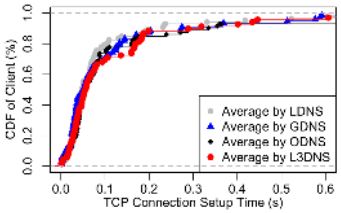

Fig. 22. Avg. end-to-end latency for Google CDNs

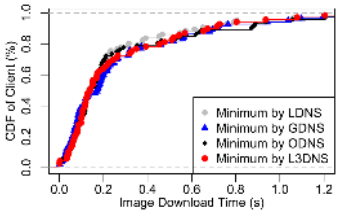

Fig. 23. Min. Image download time for Google CDNs

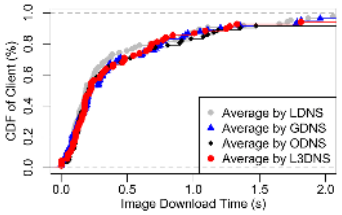

Fig. 24. Avg. Image download time for Google CDNs

D. Overall Performance Variation of CDN servers

Given that the Web performance is affected by the choice of a CDN server within a DNS resolution and by the choice of a DNS server, we now explore how these choices could impact the Web performance in combination. We argue this is important because many users opt for open and public DNS servers that offer faster resolutions and also because LDNS does not always resolve to best CDN servers, which we show later in Figure 41. In Figures 25 and 29 we compare the minimum, average, and maximum end-to-end latency between clients and the CDN servers resolved by any of the DNS servers used in our study. In these graphs we show that the minimum end-to-end latency to CDN servers, when resolutions from LDNS, GDNS, ODNS, and L3DNS are combined, is significantly lower than the average end-to-end latency. For example, in Figure 25, we show that for 90% of the clients the minimum end-to-end latency to Akamai CDN servers, resolved by any of the DNS servers, is less than 50 ms and the average end-to-end latency is more than 200 ms. We also show similar trend in end-to-end latency for Google CDNs and the image download time for Google and Akamai CDNs in Figures 29, 26, and 30. Therefore, client applications that rely on resolutions from multiple DNS providers, as proposed by Vulimiri *et al.*, may not connect with CDN servers that have the least end-to-end latency to the client [15].

Next, in Figures 27 and 31 we show the ratio of minimum end-to-end latency to maximum end-to-end latency for Akamai and Google CDN servers returned by DNS servers in our study. For example, in Figure 27, we show that for the 85% of users connecting to Akamai CDN servers, the end-to-end latency for the fastest available server is 40% lower than the slowest server. Similarly, in Figure 31, the end-to-end latency to the fastest available Google CDN server is only about 30% lower than the slowest server.

We also show similar ratios for image download time from Akamai and Google CDN servers in Figures 28 and 32. For example, in Figure 28, we show that for 90% of the users the image download time from the fastest available Akamai CDN server is only 50% faster than the slowest server. Similarly, in Figure 32, the image download time from the fastest available Google CDN server is 30% faster than the slowest server.

E. Causes of CDN Performance Variation

In previous sections, we discovered the variation in the performance of CDN servers resolved by different DNS providers. While investigating the cause of this variation, we also found that DNS resolutions often direct clients to CDN servers in different IP subnets and different network locations. For example, as much as about 45 and 40 percent of the clients were directed to CDN servers in different IP subnets when resolving Akamai and Google Web domains, respectively.

Next, we investigate the impact of inconsistency in DNS resolutions on client redirection. Due to inconsistency in DNS resolutions, some clients are often directed to servers in different geographic locations. For example, a client in Virginia was directed to servers in Cambridge, California, and Texas in three different DNS resolutions from the same DNS server. Therefore, we argue that when DNS resolutions are inconsistent, clients' Web requests may have to be served by CDN servers in locations farther than the closest available CDN server. Although this behavior is maybe due to routine load balancing, let us look at its impact on extra latency perceived by clients in connecting with CDN servers.

Our method to calculate extra end-to-end latency for each client is based on the difference in minimum and mean latency to servers in different IP subnets. In Figures 33 and 34, we show the extra end-to-end latency as perceived by clients to connect with Akamai and Google CDN servers, respectively. For example, In Figure 33 we show that for 15% of the clients that receive DNS resolutions to different IP subnets from LDNS and L3DNS for Akamai CDN domains, the extra end-to-end latency perceived by clients in every round trip is more than 100 ms and 150 ms respectively. Similarly, in Figure 34, we show that for 15% of the clients, that receive DNS resolutions to different IP subnets from LDNS and L3DNS for Google CDN domains, the extra end-to-end latency perceived by clients in every round trip is more than 15 ms and 20 ms respectively.

In Figures 35 and 36, we show the extra latency in downloading images as perceived by clients in connecting with

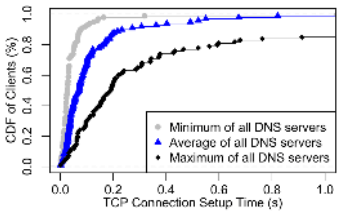

Fig. 25. Latency variation with Akamai CDNs.

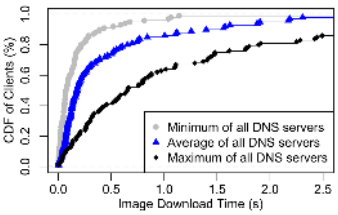

Fig. 26. Download time variation with Akamai CDNs.

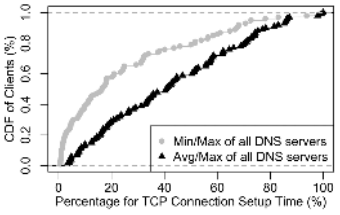

Fig. 27. Latency variation for Akamai CDNs (%).

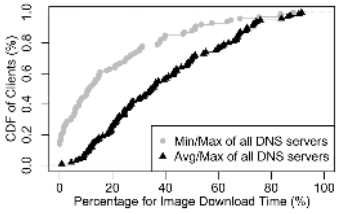

Fig. 28. Download time variation for Akamai CDNs (%).

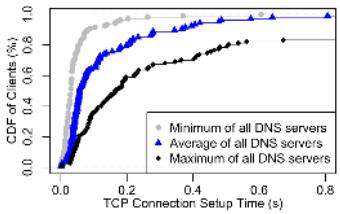

Fig. 29. Latency variation with Google CDNs.

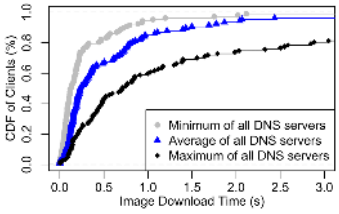

Fig. 30. Download time variation with Google CDNs.

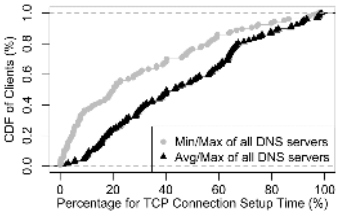

Fig. 31. Latency variation for Google CDNs (%).

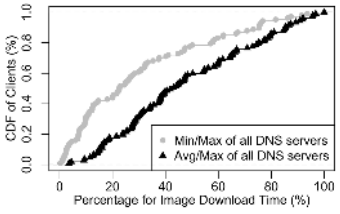

Fig. 32. Download time variation for Google CDNs (%).

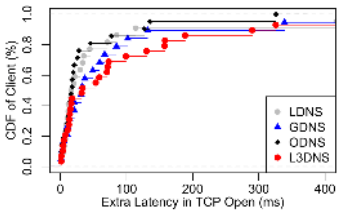

Fig. 33. Extra latency to Akamai CDNs.

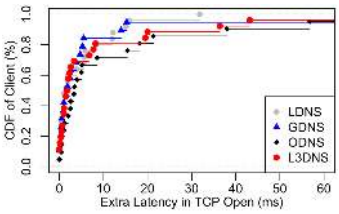

Fig. 34. Extra latency to Google CDNs.

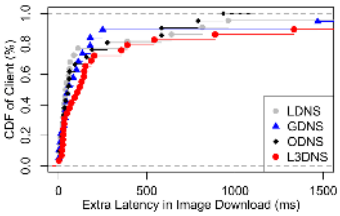

Fig. 35. Extra download time to Akamai CDNs.

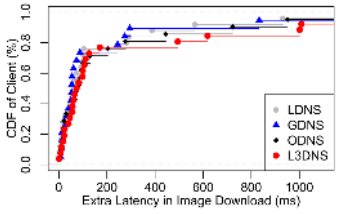

Fig. 36. Extra download time to Google CDNs.

Akamai and Google CDN servers, respectively. For example, in Figure 35 we show that for 80% of the clients, that receive inconsistent DNS resolutions from LDNS, GDNS, ODNS, and L3DNS, the extra latency in download image from Akamai CDN servers is within 100 ms, 200 ms, 300 ms, and 400 ms respectively. Similarly, in Figure 36, we show that for the same clients the extra latency in downloading an image from Google CDN servers is within 300 ms for LDNS, GDNS, and ODNS, and 500 ms for L3DNS. Therefore, if clients rely on resolutions from the regular DNS-based server selection, the penalty in terms of latency is very high. In Section III, we show how `dp` eliminates the penalty while preserving load-balancing.

F. Unreachable CDN Servers

Client devices and end-networks are often configured with firewalls to block access to some IP addresses. Automated Intrusion Detection System (IDS) software scans for any activity that might abuse the network resources. Such IDS may occasionally block access to some CDN servers, a behavior we verified in MSU's campus network. DNS providers remain unaware of such network configurations and therefore when clients request DNS resolutions, they occasionally get directed to inaccessible servers, which results in failure to start the downloads of static Web content [3][16]. Although server inaccessibility prevents the content from being downloaded all together, we believe that this could be avoided by vetting CDN servers.

Throughout our study of over a period of three months, we recorded the number of *faulty* DNS resolutions, that is, whether a DNS resolution contained at least one server address to which client could not connect. We show the number of *faulty* DNS resolutions for Akamai and Google CDNs by LDNS, GDNS, ODNS, and L3DNS for clients in different countries in Table II.

CLIENT COUNTRY	DNS	*#faulty* RESOLUTIONS	
		Akamai	Google
North America	LDNS	2	6
North America	GDNS	3	
North America	ODNS	5	3
North America	L3DNS		11
Europe	LDNS	2	
Europe	GDNS	1	1
Asia	LDNS	5	
Asia	GDNS	3	6
Asia	ODNS		4
Asia	L3DNS	1	
Australia	LDNS	2	
Africa	GDNS		3

TABLE II
FAULTY RESOLUTIONS FOR AKAMAI AND GOOGLE CDN SERVERS.

Out of the 123 DASU devices used in our study, we found that only about 15 devices in different continents received at least one DNS resolution that had at least one inaccessible CDN server. Specifically, for such clients in North America we found that these clients were connected through Comcast Cable, Florida Cable, Bright House Networks, Cox Communications, VTX Broadband, and Time Warner Cable.

We show that the problem of faulty DNS resolutions exists for both Akamai and Google CDN infrastructure. Further, to mitigate client direction from faulty DNS resolutions, the use of multiple DNS providers may not be useful since we discovered that popular DNS providers such as GDNS, ODNS, and L3DNS and as well as clients' LDNS sometimes direct clients to the same inaccessible CDN servers. In Section III, we show that `dp` eliminates this problem by making DNS resolutions aware of the performance and routing restrictions in the last-mile networks.

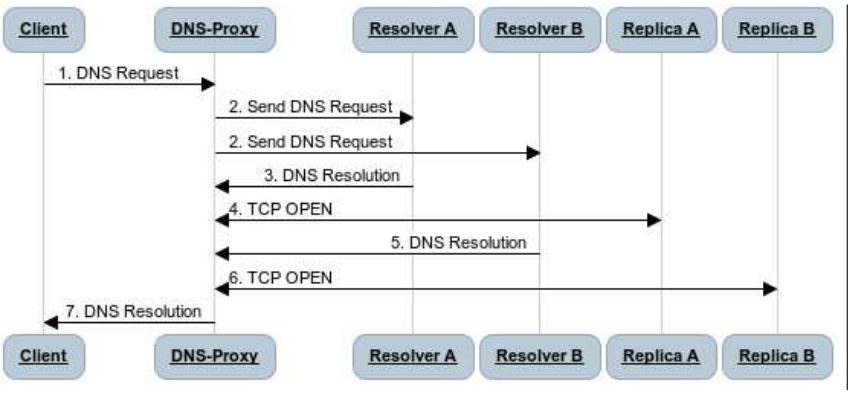

Fig. 37. dp's resolution mechanism resolution for non-cached domains.

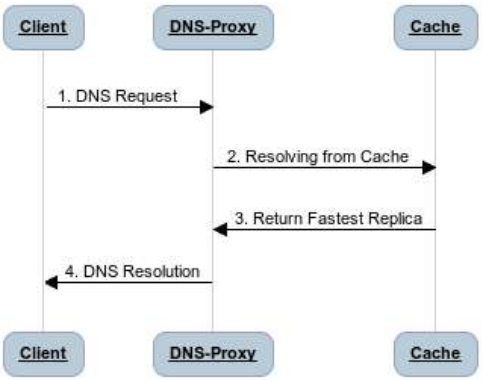

Fig. 38. dp's resolution mechanism for cached domains.

III. DNS-Proxy (dp)

Our measurement study has discovered that DNS-based load-balancing used by CDN infrastructures deployed by Akamai and Google often do not direct users to the closest CDN servers available. Therefore, we propose DNS-Proxy (dp), a client-side tool that selects best CDN servers with respect to performance of users' last mile networks. dp can also be used as a DNS server on network gateways to serve incoming DNS requests from multiple devices in the same subnet. dp runs as a virtual DNS server on client devices and generates DNS resolutions that are most suitable for the variable performance of the user's network. dp receives DNS requests from client applications and fans them out to different DNS servers. dp then probes all resolved IP addresses in parallel and returns the CDN server with the lowest end-to-end latency to the client. We make dp available as an open source tool at `http://github.com/msu-netlab/dp`.

A. Client-assisted Server Selection

We depict sequence diagrams of dp's approach to client-assisted server selection in Figures 37 and 38. dp runs on client devices and listens for incoming DNS requests on port 53, the standard port for DNS-based services. When dp receives a DNS request from a client, it forwards the request to a number of different DNS servers and waits for the DNS resolution replies. The DNS servers that the dp forwards the request to can be easily configured based on the user's preference. In our experiments, we configured dp to resolve DNS requests from LDNS, GDNS, ODNS, and L3DNS. As shown in Figure 37, after dp receives a DNS resolution, it sends TCP SYN packets, in parallel over raw sockets, to port 80 (standard port for hosting HTTP based services) and port 443 (standard port for hosting HTTPS based services) to each resolved CDN 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.

The end-to-end latency to each CDN server is measured based on the time to receive the TCP SYN/ACK packet from the probed server. dp collects the end-to-end latency to each server and maps the domain name being resolved to the CDN server with the lowest end-to-end latency. dp then returns the server with the lowest end-to-end latency identified, as a resolution for the client's DNS request. Apart from directing clients to the fastest available CDN servers, dp prevents directing clients to the servers for which it never receives a TCP SYN/ACK packet. However, if a DNS resolution contains only one IP address, dp directs the client to that IP address, regardless of it being inaccessible.

dp sends DNS responses to clients using one of two methods. dp resolves domain names either from its own cache of DNS entries (we refer it as *warm-cache*), or delays the DNS response for a domain not in its cache to probe for the fastest server on the fly (we refer it as *cold-cache*). As shown in Figure 38, when dp has a DNS entry for the domain being resolved in its cache, dp instantly replies to the clients with a DNS resolution, which also allows to reduce the overhead for name resolution [17]. However, when a DNS entry is not available in the cache, dp relies on a user configurable deadline (set to 30 ms by default) within which dp must reply the client's DNS request with the fastest identified CDN server. The user configurable deadline ensures that users' DNS requests do not have to wait if the domain being resolved is not available in dp's cache, or if probing different CDN servers take a long time. While the use of a deadline in dp predictable response times, dp also continues to probe additional CDN servers after the deadline to refine its accuracy of server selection for future requests.

dp sets the time to live (TTL) value in the DNS response for each DNS resolution generated from the *cold-cache* to two seconds. A low TTL value in the DNS response allows clients to use a resolved server (from dp's *cold-cache*) for only a short period of time before reissuing the DNS query. We expect dp to have probed all resolved CDN servers and identified the fastest available CDN server within two seconds. At this point dp will respond to the second DNS query from its *warm-cache*. However, the TTLs for DNS responses generated from the *warm-cache* are larger than two seconds, but lower than the actual TTL values present in responses from different DNS servers. Further, dp deletes DNS entries from its cache for any domain that has been cached for more than DNS TTL, which enables dp to proactively identify the best CDN server, if the performance of the previously identified best server had changed since it was last probed [18].

B. Probing Metric

Our decision to use TCP connection setup time in dp as indicative of object download time from CDN server is based on the data collected from Dasu devices. Predicting the fastest

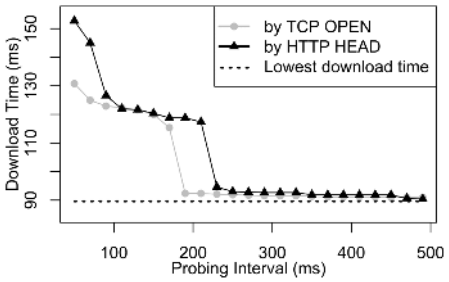

Fig. 39. TCP OPEN vs HTTP HEAD for Akamai CDNs at different probing intervals.

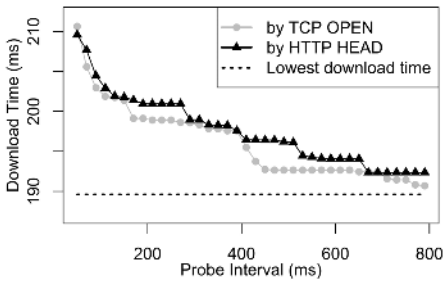

Fig. 40. TCP OPEN vs HTTP HEAD for Google CDNs at different probing intervals.

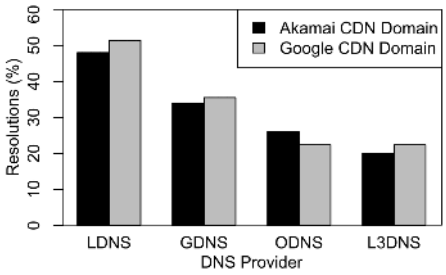

Fig. 41. Frequency of different DNS servers resolving to fastest CDN servers, as identified by dp.

available CDN server based on the actual least download time of static Web objects would reflect the true performance for a given CDN server, however, downloading Web objects from each resolved CDN server would introduce a high probing traffic and long probing delay. Therefore, we argue that the prediction of fastest available server should be based on a server selection mechanism that is faster and requires less probing traffic. We evaluate both the time to receive the first bit of HTTP response (we refer it as the HTTP HEAD method) from CDN servers and the time to establish a successful TCP connection with a CDN server (we refer it to as the TCP OPEN method).

To find an appropriate method for server selection, we compare the download time of fastest server identified by TCP OPEN and HTTP HEAD at different dp response deadlines. We refer to dp's deadlines as probing interval before step 7 in Figure 37. In Figures 39 and 40, we compare the image download time from fastest CDN servers for Akamai and Google CDN domains, as identified using the TCP OPEN and HTTP HEAD methods. The x-axis shows dp's probing interval following a client DNS request. The y-axis shows the average of image download times from the fastest CDN servers identified at different probing intervals of dp. The dashed horizontal line shows the download time from the fastest CDN server averaged over all clients, regardless of the server selection method and dp's probing interval.

We show that as the probing interval increases dp continues to listen for additional resolved CDN servers to refine accuracy of client-assisted server selection for future requests. We also note that 1) At any given probing interval, the TCP OPEN method is more accurate on average than the HTTP HEAD method, because TCP OPEN receives higher percentage of probes back in a given interval, and 2) the line representing TCP OPEN method tends towards the dashed line, which indicates that the download time of the server chosen by the TCP OPEN method is approximately that of the server with the least image download time. For example, in Figure 39, the image download time of the fastest CDN server identified by the TCP OPEN method at the probing interval of 180 ms is 90 ms, whereas, the image download time of the fastest server identified by the HTTP HEAD method is 120 ms at the same probing interval.

Although probing may introduce an extra load on the client's network, it is important to note that dp caches probe results to avoid probing same servers in subsequent requests. Further, since dp reduces the number of DNS requests leaving the network or the client device by resolving them from it's cache, the overall network load is reduced. Finally, our evaluation of dp on a 24-hour DNS trace collected from the MSU's network shows that the network traffic sent and received by dp is less than 700 KB for every 500 resolved Web domains. We argue that in comparison to the benefits dp brings for clients, the dp probing traffic is reasonably negligible.

IV. Results

Next we demonstrate, based on extensive measurement of dp, that client-assisted server selection is more effective at identifying fastest available CDN servers and reducing webpage load times across CDNs, last mile networks, and geographic locations, than the current DNS-based server selection techniques.

A. Identifying DNSs with Fastest CDN Servers

In Figure 41, we show whether the fastest CDN server chosen by dp was resolved from LDNS, GDNS, ODNS, or L3DNS. We show that for resolving Akamai CDN domains, only 48% of resolutions from LDNS contained the fastest CDN server, whereas, while resolving Google CDN domains, 51% of resolutions from LDNS contained the fastest CDN server. We also discovered that resolutions from different DNS providers may both contain one or more common IP addresses, which were identified as the fastest server by dp in some resolutions. For such resolutions, we increment the height of bar for each DNS provider that contained the fastest CDN server, which is indicative of the fact that the sum of heights of the bar plots do not aggregate to 100%. Since none of the DNS servers used in our study are reliable in directing clients to the closest available CDN replicas at all times, we argue that dp provides a complementary approach to DNS-based server selection by relying on multiple DNS providers and directing the clients to the fastest available CDN replicas from resolved CDN addresses at all times.

B. Faster Web through DNS-Proxy

We compare the webpage load time of websites hosted on Akamai, Google, Level 3, CloudFront, and Reflected Networks CDN servers, when clients use their ISP-provided LDNS and dp. We list the website addresses, number of Web objects, the hosting CDN, and the total page size in Table III. We configured experiments on devices in California (CA), Montana (MT), Illinois (IL), and New York (NY) to load these websites 20 times each from CDNs resolved by LDNS and dp. To prevent object loading from browser cache, we loaded websites in Google Chrome browser's incognito window.

Webpage	Host CDN	# Web objects	Page Size
huffingtonpost.com	Akamai	374	3.4 MB
developer.android.com/tv/	Google	60	6.8 MB
level3.com	Level 3	58	1.4 MB
chictopia.com	CloudFront	90	1.8 MB
an adult "tube" site	Reflected N/w	70	7.6 MB

TABLE III
DETAILS OF WEBPAGES LOADED FOR COMPARISON.

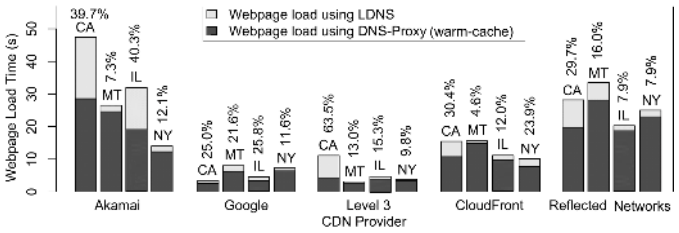

Fig. 42. Comparion of Webpage load times using LDNS and DNS-Proxywith *warm-cache*.

In Figure 42, we show the average webpage load time of different websites using LDNS and dp with *warm-cache* from CA, MT, IL, and NY. Above every bar, we show the reduction in webpage load time in percentage achieved when using dp. dp speeds up page loads across CDN providers and geographic locations. Specifically, for static content heavy website, such as *huffingtonpost.com*, dp provides a speedup of as much as 40%. For other websites with relatively fewer or smaller images, dp provides speedup close to 30% in the common case.

In Figure 43 we compare the average load times of individual Web objects, hosted on different websites, when loaded from servers resolved by LDNS and dp with *warm-cache*. We loaded a total of 35443 Web objects hosted by different CDN providers. We show the average object load time (which includes the DNS lookup time, TCP connection setup time, time to receive the first bit of HTTP response, and time to download the object) using LDNS and dp and label each bar with the average percentage reduction in individual object load time achieved by dp as compared to LDNS. We show that even when dp does not have a resolution for a domain in its cache, dp is effective in reducing the webpage load time by delaying DNS responses to identify fastest CDN servers. For example, dp reduces the load time of each web object hosted on CloudFront CDN servers by about 43%. For web objects hosted on other CDNs, dp reduces the load time for each object by about 20% on average.

Finally, in Figure 44, we compare the average of 20 webpage load times each from CDNs resolved by using LDNS and dp with *cold-cache*. For this comparison study, we configured dp's DNS resolution deadline to 30 ms and cleared dp's cache after each page load. We show that even when dp does not have a DNS resolution for a Web domain available in its cache, delaying DNS resolutions by 30 ms to identify faster servers helps to reduce the overall webpage load time. We show the average percentage reduction in webpage load times using dp with *cold-cache* on top of each dp bar in the graph. The one exception in our data is for webpages hosted on Google CDNs for which the average webpage load time with dp's *cold-cache* was marginally higher than the average load time with LDNS. We believe that in most cases dp could eliminate the *cold-cache* penalty through dynamic adjustment of DNS response deadline or to not delay DNS responses for domains where dp shows no gains for overall webpage load time – a subject of our future work.

V. DISCUSSION

Two possible concerns come to mind with widespread adoption of dp. First, would dp probing introduce higher loads on CDN servers thereby increasing queuing delays? Because dp uses the light-weight TCP OPEN process and immediately closes the connections after they are established, we believe that the increase in network load, or CDN server resources due to probing is not significant.

Second, does client-based server selection disregards the existing DNS-based load balancing? Because dp selects CDN servers only from among the set resolved by DNS infrastructure, CDNs retain control over which replica server a client may connect to. As a result dp clients in different network locations will probe different sets of CDN servers. While it is possible that clients in the same location might select the same fastest CDN server and potentially increase its queuing delays, we plan to extend dp to avoid this situation by switching to HTTP HEAD-based probing, which takes server queuing delays into account.

Although we have demonstrated the benefits of dp in accelerating Web services hosted on CDNs, our method is applicable to other replicated server selection problems. To facilitate wide dp deployment we plan on integrating our method with bind DNS software commonly used in many last mile networks. Currently we make a standalone implementation of dp available at http://github.com/msu-netlab/dp.

VI. RELATED WORK

In spite of several years of efforts to reduce network latency, CDNs and content providers strive to deliver a responsive experience to their users. In addition to evolving DNS-based server selection for Web applications, CDN server selection techniques for improving the delivery of live video have also been explored as an alternative to DNS-based load balancing [19][20]. Although, server selection for video and Web content delivery are related to each other, we believe that the techniques to accurately identify the best CDN servers for both of these services need to consider different metrics (latency vs. available bandwidth), because of difference in application requirements. Therefore, we only discuss studies that aim to improve server selection for Web applications.

A. Reducing DNS Lookup Time

Vulimiri *et al.* proposed a client-side tool for sending DNS requests to multiple DNS servers and using the first received DNS resolution on the client [15]. CoDNS is a similar tool that distributes incoming DNS requests to multiple DNS servers to mask the delay in DNS lookups [21]. However, in Figure 18, we show that different DNS servers may direct clients to CDN servers with different end-to-end latencies. Therefore, directing clients to server in the first DNS resolution may not always direct clients to the closest available CDN server.

Shang *et al.* proposed a tool to reduce the DNS cache miss rate by exploiting similarity of requested Web domains to the domains that already have a DNS resolution cached [22]. DNS

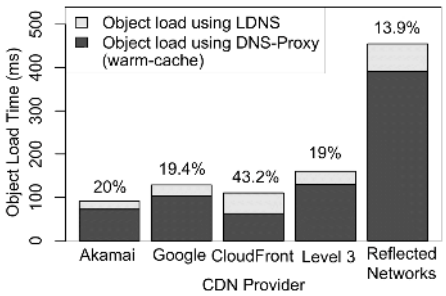

Fig. 43. Comparison of Web object load times using LDNS and DNS-Proxy with *warm-cache*.

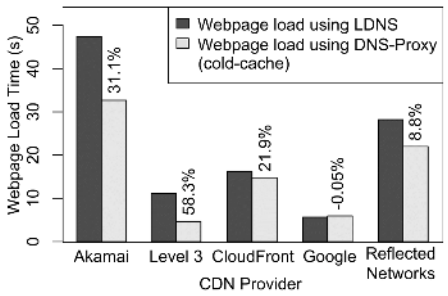

Fig. 44. Comparison of Web object load times using LDNS and DNS-Proxy with *cold-cache*.

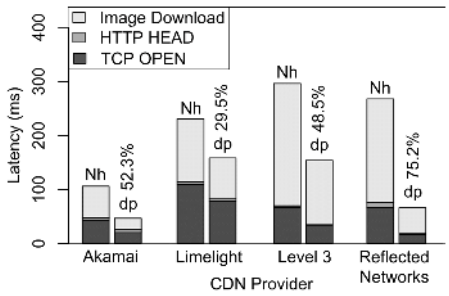

Fig. 45. Comparison of object load times using CDN servers resolved by dp and Namehelp.

Pre-Resolve is another technique to eliminate DNS lookup delay by resolving domain names proactively during Web page rendering [23]. Although these two techniques reduce the impact of DNS on webpage load time, they do not consider performance variation between CDN servers, nor accuracy of resolutions from different DNS servers.

B. DNS Server Selection from End-devices

Namehelp is a client-side tool to identify the DNS server that (on average) directs clients to the fastest available CDN servers [24]. Namehelp's mechanism to identify the best DNS server relies on user's Web browser history, in that, Namehelp resolves Web domains (accessed by the user in the past) from multiple DNS providers. Based on the average performance of CDN servers returned from different DNS servers for Websites used in the past, Namehelp configures the client's default DNS server to the DNS server that directed the client to servers with least latency on average. Namehelp is similar to dp in that it measures the client's latencies to multiple CDN servers resolved by different DNS servers for client-side server selection. However, we illustrate several differences in server selection techniques used by dp and Namehelp, and also show that dp direct clients to servers remarkably faster than servers resolved by Namehelp, on average.

- In our experience with Namehelp we discovered that in order to identify the best DNS server for a client, Namehelp sends over 27000 DNS requests (of size 80 bytes each) to different DNS servers every 15 minutes, followed by receiving a DNS response (of size 150 bytes each) for each request, and finally sending an HTTP HEAD request (of size 120 bytes each) to every resolved server. We argue that, unlike dp, Namehelp creates an undesired load on the last-mile network, which could potentially impact the performance of other applications running on the device.
- To improve the accuracy of server selection, Namehelp sends an HTTP HEAD request to every resolved server. However, based on our experience with dp we show in Figure 39 that using TCP OPEN delay is more accurate indicator of server performance and generates lesser probing traffic than sending HTTP HEAD requests.
- Unlike dp, Namehelp does not compare client's latencies to different CDN servers within a DNS resolution, which is important for minimizing the end-to-end latency between clients and servers (as discussed in Section II-B).
- As shown in Figure 41, none of the DNS servers used in our study directed clients to servers with least end-to-end latency at all times. Therefore, we argue that the Namehelp's technique to resolve Web domains from an identified best DNS server (instead of identifying the best CDN server for each Web domain being resolved) might not ensure that clients will always be directed to fastest available CDN servers for all Web domains.
- Finally, unlike dp, Namehelp's resolution technique does not have a deadline within which it must resolve the requested Web domain.

In light of these differences between Namehelp (Nh) and dp, in Figure 45 we compare the average time to establish a TCP connection, time to receive the first bit of HTTP response, and the image download time from servers resolved by Namehelp and dp. For this comparison study, we configured clients to use Namehelp and dp one-by-one as their default DNS servers within a short time period. Next, we opened CDN URLs hosted on Akamai, Limelight, Level 3, and Reflected Networks CDNs on Google Chrome browser's incognito window (to prevent object loading from cache). We first resolved the Web domains from the configured DNS server, followed by sending an HTTP GET request to download an image of size 82 KB, 53 KB, 207 KB, and 32 KB from the resolved Akamai, Limelight, Level 3, and Reflected Networks CDN servers respectively. We show the average percentage reduction in the overall object load time using dp on top each dp bar. For different CDN providers, we show that from an average of 25 DNS resolutions from Namehelp and dp each, the time to open TCP connection (TCP OPEN), time to receive first bit of HTTP response (HTTP HEAD), and image download time from servers resolved by dp are about 50% faster than the servers resolved by Namehelp on average.

C. CDN Load-balancing Techniques for Server Selection

A study by Shaikh *et al.* has shown benefits of providing client's IP address in the DNS request to Authoritative DNS servers, to allow clients to connect with nearby servers [17]. EDNS protocol has similar motivation in that it also enables CDN providers to direct users to nearby servers based on the client's IP subnet [13]. However, EDNS approach is still limited by how many DNS providers and ISPs support requests with EDNS [14].

A study by Kangasharju *et al.* compares the performance of different DNS redirection techniques, such as full redirection and selective redirection, used by CDN providers to reduce the

impact of network latency on Web applications [25]. Their study shows that full redirection has superior performance over selective redirection, since selective redirection has an overhead to maintain which CDN server has what content in its cache, which may not be up-to-date and accurate at all times.

D. Relative Network Positioning for Server Selection

Previous studies have investigated the benefits of using network coordinate systems (NCS) to estimate the network latency between arbitrary end hosts [26][27][28]. Such NCS techniques also enable CDN providers to estimate the network latency between clients and CDN servers to increase the accuracy of DNS-based server selection techniques [29]. However, a study by Choffnes *et al.* shows that network coordinate systems are often not accurate when used on edge networks [29].

CDN-based Relative Network Positioning (CRP), a tool by Su *et al.* shows that clients could be directed to closest CDN servers by comparing the cosine similarities between clients and different available CDN servers [30]. However, our other recent work on server selection shows that CRP technique is often not accurate in predicting the closest servers for clients [31].

VII. Conclusions

Web application performance is affected by DNS resolutions to distant CDN servers. Although, DNS-based server selection may often direct clients to nearby CDN replicas, we show that current techniques could be improved to speed up the delivery of content. Therefore, we argue that clients are best positioned in the network to choose closest CDN servers. We propose DNS-Proxy (`dp`), a client-side tool that probes each resolved CDN address and directs clients to the fastest available servers. Effectively, `dp` shares load balancing functionality with CDNs by selecting from a set of resolved servers. Our measurement study on CDN infrastructure deployed by five major CDN providers shows that `dp` reduces webpage load time by 29% on average. If `dp` has already resolved a Web domain, the reduction in webpage load time is as much as 40%. Finally, `dp` reduces the download time of individual static Web objects by as much as 43%. Overall we believe `dp` enables a more effective use of existing CDN infrastructure and represents a complementary strategy to a continual increase of geographic content availability.

Acknowledgements

The authors would like to thank Mario Sanchez for his help in setting up the experiments on Dasu, and Ajay Miyyapuram and Kanika Shah for suggested improvements to an early version of this manuscript. Further, the positions, strategies or opinions reflected in this article are those of the authors and do not necessarily represent the positions, strategies or opinions of Akamai.

References

[1] 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.

[2] 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.

[3] R. L. Burt, "Why are images not loading?." https://www.facebook.com/help/community/question/?id=10100214862890089&ref=notif&notif_t=answers_answered, Jun. 2013.

[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] I. Grigorik, "Latency: The New Web Performance Bottleneck." https://www.igvita.com/2012/07/19/latency-the-new-web-performance-bottleneck/, Jul. 2012.

[6] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall, "Demystify page load performance with wprof," in *USENIX NSDI*, Apr. 2013.

[7] 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.

[8] M. Belshe and R. Peon, "SPDY Protocol." http://tools.ietf.org/html/draft-mbelshe-httpbis-spdy-00, Feb. 2012.

[9] "Everything You Need To Know About CDN Load Balancing." http://www.webtorials.com/main/resource/papers/Dyn/paper1/CDN-LoadBalancing.pdf, Sept. 2014.

[10] "Traffic Director." http://dyn.com/traffic-director/, Sept. 2014.

[11] A. Barbir, B. Cain, R. Nair, and O. Spatscheck, "Known Content Network (CN) Request-Routing Mechanisms." https://tools.ietf.org/html/rfc3568, Jul. 2003.

[12] 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.

[13] S. Souders, "Extension Mechanisms for DNS (EDNS(0))." http://tools.ietf.org/html/rfc6891, Apr. 2013.

[14] J. P. Rula and F. E. Bustamante, "Behind the Curtain - Cellular DNS and Content Replica Selection," Nov. 2014.

[15] A. Vulimiri, P. B. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Low Latency via Redundancy," in *ACM CoNEXT*, Dec. 2013.

[16] V. N. Padmanabhan, S. Ramabhadran, S. Agarwal, and J. Padhye, "A Study of End-to-end Web Access Failures," in *ACM CoNEXT*, Dec. 2006.

[17] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of dns-based server selection," in *IEEE Infocom*, Apr. 2001.

[18] C. Pelsser, L. Cittadini, S. Vissicchio, and R. Bush, "From Paris to Tokyo: On the Suitability of Ping to Measure Latency," in *ACM IMC*, Oct. 2013.

[19] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "A Case for a Coordinated Internet Video Control Plane," in *ACM SIGCOMM*, Aug. 2012.

[20] M. K. Mukerjee, J. Hong, J. Jiang, D. Naylor, D. Han, S. Seshan, and H. Zhang, "Enabling Near Real-time Central Control for Live Video Delivery in CDNs," in *ACM SIGCOMM*, Aug. 2014.

[21] K. Park, V. S. Pai, L. Peterson, and Z. Wang, "CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups," in *USENIX OSDI*, Dec. 2004.

[22] H. Shang and C. E. Wills, "Piggybacking Related Domain Names to Improve DNS Performance," *Computer Network*, vol. 50, Aug. 2006.

[23] G. Developers, "Pre-Resolve DNS." https://developers.google.com/speed/pagespeed/service/PreResolveDns, Apr. 2015.

[24] J. P. Rula and F. E. Bustamante, "Namehelp Mobile." http://aqualab.cs.northwestern.edu/projects/237-namehelp-mobile, Aug. 2014.

[25] J. Kangasharju, K. W. Ross, and J. W. Roberts, "Performance Evaluation of Redirection Schemes in Content Distribution Networks," in *Computer Communications*, 2000.

[26] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A Decentralized Network Coordinate System," ACM SIGCOMM, Sept. 2004.

[27] T. S. E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches," in *IEEE INFOCOM*, Apr. 2001.

[28] B. Wong, A. Slivkins, and E. G. Sirer, "Meridian: A Lightweight Network Location Service Without Virtual Coordinates," ACM SIGCOMM, Aug. 2005.

[29] D. R. Choffnes, M. A. Sanchez, and F. E. Bustamante, "Network Positioning from the Edge: An empirical study of the effectiveness of network positioning in P2P systems," in *IEEE INFOCOM*, Mar.

[30] A. jan Su, D. Choffnes, F. E. Bustamante, and A. Kuzmanovic, "Relative Network Positioning via CDN Redirections," in *ICDCS*, Jun. 2008.

[31] S. Micka, U. Goel, H. Ye, M. P. Wittie, and B. Mumey, "Internet Latency Estimation Using CDN Replicas," in *IEEE ICCCN*, Aug. 2015.