Bringing Local DNS Servers Close to Their Clients

Hangwei Qian
Case Western Reserve University
Cleveland, Ohio, USA

Michael Rabinovich
Case Western Reserve University
Cleveland, Ohio, USA

Zakaria Al-Qudah
Dept. of Computer Engineering
Yarmouk University, Jordan

***Abstract*—This paper provides an indication that the distance between clients and their local DNS servers (LDNS) can have a significant negative impact on the performance of content delivery networks (CDNs). Consequently, we propose a novel peer-to-peer client-side DNS mechanism that moves LDNS close to their clients while still allowing nearby clients to share the common DNS cache. Through trace-driven simulations and prototype testing, we show that our approach holds significant promise of facilitating better server selection by CDNs.**

I. Introduction

DNS-based request routing is widely used to facilitate transparent server replication in today's large-scale Internet platforms. For example, content delivery networks (CDNs) such as Akamai and Limelight, as well as large-scale content providers such as Google, use DNS-based request routing to forward client requests to nearby content replicas. The same mechanism is also used in cloud computing platforms, such as Google AppEngine.

DNS-based request routing [1] utilizes the need for Web clients to execute a DNS query to resolve the server's hostname before performing the rest of the interaction. When the authoritative DNS server receives this query, it responds to the client's DNS query with the IP address of a server that is dynamically selected to be "close" to the originator of the query (where the meaning of "close" is at the discretion of the service provider, and is often part of its core expertise). The subsequent interaction of the client occurs with the nearby server, resulting in better performance. However, the authoritative DNS server only sees the IP address of the client's DNS server (LDNS - for "local DNS") and not the client itself, when making the server selection decision. Thus, the effectiveness of this mechanism is inherently limited by how close the clients are located to their LDNS.

The proximity between clients and their LDNS was studied in [2], [3]. In particular, [2] found that only 16% of the clients were in the same *network-aware cluster* [4] with their LDNS servers, and about 36% of them were not in the same autonomous system. These mismatches can negatively affect the effectiveness of DNS-based server selection. An obvious way to avoid this problem would be to run the LDNS component on every host, but that would negate the advantage of a shared cache provided by current distinct LDNS configurations. The latter is important because the LDNS cache not only reduces the burden on the DNS infrastructure, but also improves the response time of the DNS queries satisfied from the cache. An alternative approach suggested in [3], which is to carry the IP address of the client requesting the name resolution in the DNS query message, so that the exact location of the client is conveyed to authoritative DNS, similarly negates the shared LDNS cache.

In this work, we propose to run a LDNS component on every host, and at the same time use peer-to-peer (P2P) techniques to share cached DNS resolutions among hosts within the same subnet. In our proposed system, peers sharing the same gateway would form a P2P system, where different peers would be responsible for different domain names according to the distributed hash mapping utilized by the P2P infrastructure. Since all peers in the P2P system are with the same gateway, they are supposed to be close to each other. Also, by sharing resolutions among the peers, we can preserve most of cache benefits of the current system. Through our experiments in section V, we show that the overhead of running the P2P and LDNS components on each host is trivial, and the P2P lookup delay for DNS resolution is acceptable. At the same time, sharing DNS resolutions among proximal neighbors promises clients significant improvements in terms of network delay and bandwidth of their interactions with replicated content servers.

Previously, Cox et al. [5] and Ramasubramanian and Sirer [6] discussed P2P-based mechanisms for the whole DNS infrastructure. Our work does not affect the entire DNS infrastructure and instead only changes the local LDNS setup. CoDNS [7] is client-side P2P approach but, unlike our scheme, it focuses on reliability and performance of DNS without addressing the client-to-LDNS proximity.

II. Motivation

To get a sense of the importance of the problem we are addressing, we provide an indication of the effect of the separation between clients and their LDNS on server selection by content delivery networks, which carry a significant portion of the entire Web accesses.[1] In a parallel project, we use a technique from [2] to instrument a popular Web site and harvest provably correct associations between Web clients and their LDNS servers. We have collected nearly 161K distinct client/LDNS pairs and utilized a GeoIP database [8] to map them to geographical locations. After removing the pairs where at least one of the hosts in a pair could not be

This work was supported in part by the National Science Foundation under grants CNS-0831821 and CNS-0721890.

[1]Indeed, Akamai alone claims to deliver 20% of the total Web content, the claim that our parallel ongoing study of our university traffic finds conservative.

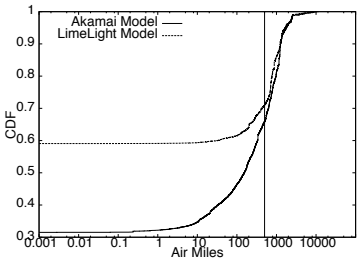


Fig. 1. Extra air miles between clients and CDN servers due to LDNS-based server selection

mapped, we ended up with nearly 154K mapped pairs. We then modeled server locations for the two leading CDNs - Akamai and Limelight - and estimated how much off their server selection would be for the clients in our dataset from the perspective of geographic proximity. Obviously, CDNs use more sophisticated considerations for server selection, but our back-of-the-envelop estimation indicates the impact of the issue we are considering.

We model CDN server locations as follows. For Akamai, which is known to have presence in most sizable cities in the world, our model consists of 1033 server locations, with 754 major cites in the US and 279 major cities in the rest of the world (Akamai claims around 7000 locations in total). On the other hand, Limelight has only a couple dozen data centers and we modeled their actual data center locations as reported in [9]. In either case, we select the closest CDN server based on the client's location (which would be the desired choice according to geographic proximity) and the closest CDN server based on the location of the corresponding LDNS (which would be the currently practical choice for CDNs). Then we obtain the extra miles the client's packets need to travel because of the potentially wrong choice of the CDN server.

Figure 1 shows the cumulative distribution function (CDF) of the extra miles. As expected, more Akamai clients incur distance penalty at low penalty values: Akamai's numerous server locations allow for finer-grained geographical server selection but by the same token make it more sensitive to the inaccurate representation of client's location by its LDNS server.[2] Most important, however, is that roughly 30% of all clients incur over 500 miles extra travel for both models. Furthermore, our results confirm an early finding from [2] that many clients reside in an autonomous system that is different from that of their LDNS servers - we had around 21% of such clients in our dataset.

These results indicate that a significant portion of clients are configured with LDNS far enough to adversely affect proximity-based CDN server selection. In principle, there could be a variety of reasons for this phenomenon, such as: (i) ISPs deploy a relatively small number of LDNSs in their networks to save administration and equipment costs; (ii) LDNSs are not uniformly distributed in ISP's networks, even if their overall number might be large; (iii) when configuring the LDNS servers for clients, ISPs do not specifically attempt to reduce the proximity between clients and the LDNS servers assigned to them; and (iv) ISPs assign a large number of clients to the same shared LDNS server in an attempt to increase its cache utilization, and a large population of clients stipulates assembling them from a large geographical area.

Although all these reasons are theoretically possible, we speculate that reason (i) is the most likely. Indeed, previous work (supported by our results below) indicates that very high DNS cache hit rates are achieved with modest number of clients sharing the LDNS cache [10]. We can discount other reasons by assuming that ISPs would be willing to improve the service for their clients when it can be done with little effort on their part.

Given the above reasoning, one potential solution for the ISPs is to add LDNS functionality to the boxes that are deployed more widely, such as access routers at the points of presence. For example, each access router can run the LDNS software and all the clients sharing the same access router would be configured to use it for DNS resolutions. Since PoPs are supposed to be close to their clients, they are well positioned to act on behalf of their clients for server selection. However, this approach would require ISPs to customize and redeploy these boxes in their networks, which may incur high replacement cost. Furthermore, these boxes are often highly optimized specialized pieces of equipment, and loading them with significant new functionality may not be feasible.

Another way to address this problem is proposed in [3]. Authors propose to carry the IP address of the client requesting the name resolution in the DNS query message. In this case, the authoritative DNS server would know the actual identity of the clients and make server selection based on the identity of the actual client and not its LDNS server. While this method indeed makes DNS-based server selection precise, it requires the modification of the existing DNS protocol. Furthermore, it impedes sharing of the returned DNS resolutions among clients since responses can be specific to a particular client. This can greatly diminish the DNS cache effectiveness (as we show in Section V, the hit ratio for a single client is only about 60%).

In our proposed system, clients themselves run an LDNS component and at the same time use peer-to-peer (P2P) techniques to share cached DNS resolutions with their neighbors. This approach requires no modifications to the DNS protocol and minimal effort/investment on the part of the ISPs. In the remainder of the paper, we describe the architecture and implementation of our system and present a preliminary evaluation study of its performance.

III. Architecture

This section outlines the architecture of our system. We also describe several issues and our design decisions to address them.

[2]This should not be construed as Limelight's advantage because this penalty represents the extra distance over the distance to closest server and not the absolute distance between the clients and CDN servers.

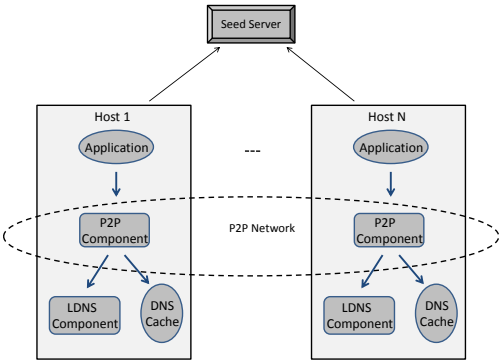


Fig. 2. System architecture

A. Overview

As shown in Figure 2, every host in our architecture runs a LDNS component and a P2P component. When a host starts up, in addition to the gateway and other information the host normally gets from the DHCP server, the host will also get the IP address of the *seed server* that is maintained by the ISP and which keeps track of various client clusters. The members of each cluster form a P2P system within their cluster. The host sends the gateway IP address to the seed server, the latter selects a client cluster for the host to join based on the gateway information and returns *seeds* (a few hosts that are already members of the selected cluster) to the host. Then the P2P component on the host sends a JOIN message to one of the seeds to join the P2P system of the corresponding cluster. In selecting clusters for each new client, the seed server ensures that clusters comprise only clients that are topologically close to each other.

When an application at host i , e.g., web browser, needs to resolve a domain name, the application sends the DNS query to its co-located P2P component, which sends a LOOKUP message into the P2P system. The lookup message is routed through the cluster according to the distributed hash table employed by the P2P network and using the requested domain name as the key, until it arrives to the destination host k, which is the host responsible for the given domain name. The P2P component on host k asks its LDNS component to perform an actual DNS query (in the same way as current LDNS servers do). After obtaining the resolution, host k puts it in a RESPONSE message and sends it to host i. The P2P component at host i then returns the resolution to the application.

P2P sharing of DNS resolutions allows hosts to benefit from a shared DNS cache. Each host maintains a local cache of DNS responses, whether obtained previously by its own applications or as a result of processing a lookup message from a peer. Before sending a lookup message, the P2P component checks its local DNS cache and responds to the application immediately if the requested DNS record is available there and is still valid. Similarly, the destination peer of a lookup message checks its local DNS cache and uses the cached DNS record to construct its RESPONSE message if the valid DNS record is available in its local cache.

Finally, the internal reliability mechanisms of P2P networks, which are designed for frequent peer disconnections, provide for continued operation of our system in the face of unreliable peers. In particular, our prototype uses Bamboo, a P2P system designed to tolerate high peer churn [11].

B. Traversing Firewalls and NAT Boxes

Network address translation (NAT) devices present two challenges to our system[3]. First, a host behind a NAT box obtains a private IP address of the NAT box (e.g., home router) as its gateway during startup. While this is acceptable in most organizational settings, where the entire P2P cluster can be formed behind the firewall using private IP addresses (and where our system may in fact not be needed if the organization runs its own LDNS within its local network, in which case hosts would be already close to their LDNS), residential clients may need to form clusters that span multiple home networks. In this case, the host must provide the IP address of its ISP access router, and not of its home router, to the seed server.

To deal with this issue, we note that home routers obtain the IP address of their ISP access router by executing their own DHCP protocol over the public side of the network. Furthermore, most home routers (e.g., linksys) provide an API (usually HTTP-based) to the hosts on the private side that allows hosts to obtain this information as an HTML page. Thus, the host can parse the IP address of its ISP access router from the HTML text and submit it to the seed server.

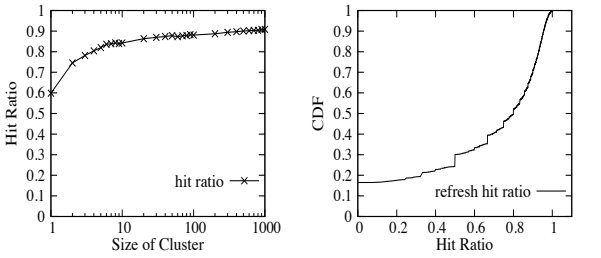
The second issue is typical for all P2P networks and has to do with NAT boxes preventing a peer from being contacted from outside hosts. We use standard techniques to address this issue. The NAT boxes can be configured with a permanent port mapping allowing outside communication (in fact, this can be done by a script using the NAT's API). The closed nature of our P2P network alleviates attendant security issues.[4]

C. Degenerate Clusters

Given that many clusters in our envisioned system would be residential, their size can drop significantly during periods of low activity, e.g., at night when home users may turn off their computers. When this happens, as shown in section V, the DNS hit ratio can degrade significantly. While this would have limited effect on the load on the DNS infrastructure because the hit rate reduction would be canceled by the overall low level of use, the users may experience higher response time in their Internet accesses. We considered two ways to address this problem: (i) merge extremely small clusters with other nearby clusters - in this case hosts in degraded clusters would query

[3]Firewalls present the same issues and our discussion applies to them too.

[4]Furthermore, permanent port mappings and the presence of the seed server as coordinator allow outside communication to be permitted on demand only: the seed server alerts two peers that must establish a link; the peers send each other join messages, each of which opening a hole in the sender's NAT box. The second message to arrive at the destination peer will find the hole open and establish the communication channel, which can then be maintained with periodic keep-alive messages.



(a) Hit rate vs. cluster size (b) Per-client hit rate with refresh

Fig. 3. DNS hit rate

the seed server for a nearby cluster to join, and (ii) let the P2P component *proactively refresh* expired DNS resolutions when cached records become invalid (as suggested in [12] in a different context), with the additional DNS load, again, being counterbalanced by the overall low activity level.

The tradeoffs between these approaches are that, with cluster merging, a degraded cluster may not find a sufficiently close neighbor cluster to join. At the same time, the efficacy of proactive DNS refresh depends on how much clients revisit old domain names. Our experiments of Section V show that roughly half the clients get over 80% DNS hit rate with DNS pre-refresh even when the cluster reduces to a single peer. We also note that the effectiveness of cluster DNS cache sharing increases rapidly with the cluster size. As we will see, a cluster with only six hosts has DNS cache hit rate of at least 85% even without cache refreshing. Thus, we consider handling of degraded clusters as more of a precautionary mechanism.

IV. Prototype Implementation

We implemented our system and deployed it both in our lab and the Emulab testbed. Our implementation uses bind-9.5.0 [13] as the LDNS component. We built our P2P component on top of Bamboo DHT [11] table. In the prototype, we modified Firefox to send DNS queries directly to the P2P component instead of LDNS. In real implementation, all the components of our system would be wrapped in a daemon listening on port 53 and mimicking a regular LDNS server. Then, to route local DNS queries from all of the host's applications to our P2P component, each host would simply be configured to use localhost as its LDNS server.

V. Evaluation

To estimate the performance of our approach, we first evaluate the DNS hit ratio in client clusters of various sizes. Then we evaluate the overhead, in terms of CPU and memory, for running the P2P component and DNS component on the host. Next, we measure the delay for looking up DNS resolutions in the P2P system in client clusters of varying size. Finally, we measure the end-to-end improvement when employing our system.

A. Hit Ratio

We use trace-driven simulation to explore DNS hit rates in our system. We used an anonymized 4-hour HTTP trace collected on May 9, 2006 at Case Western Reserve University. The trace comprises over 5.8 million HTTP downloads from over 6,600 clients. We extracted the hostnames from the trace and obtained TTLs of the corresponding type-A DNS responses by performing actual DNS queries for each name. When evaluating the hit ratio of a client cluster with a specific size s, we randomly choose s clients from the trace and simulate each access from these clients while mimicking the DNS cache behavior, which we can do faithfully by using the actual TTLs obtained. We repeat each simulation 20 times for different random client clusters and report the average hit ratio.

Figure 3(a) shows the DNS hit rates for different sizes of client clusters. The results (which are consistent with the previous findings from [10]) show that even small client clusters exhibit high DNS hit rates: as long as the size of client clusters is bigger than 6, the difference of hit ratio is within 7% of the maximum possible, when the cache is shared among all the clients in the trace.

Our next experiment considers hit rates achievable by degenerate clusters with DNS refresh discussed in Section III. We consider the extreme case of clusters with only one host and obtain the DNS hit rate of each client in the trace assuming every DNS record in the cache is refreshed upon expiration and hence available to the client when requested. We did not evaluate the extra DNS traffic this would cause as this mechanism would only be engaged during periods of low activity. Figure 3(b) shows the CDF of hit ratios with refresh for all clients. According to the figure, roughly half the clients would have 80% or larger hit ratio. Still, many clients would have low hit rate, with almost 20% of clients exhibiting under 30% hit rate. We note, however, that this result is conservative because the cache contents of individual client are limited by the short duration of our trace.

B. Resource Consumption

In this subsection, we evaluate the resource consumption of the new components on the client machines, focusing on the CPU and memory. We deployed our prototype at Emulab [14] for this experiment. To overcome the difficulty of scheduling a large number of machines, we use the following setup. We allocate 28 physical machines, each with 2 core CPU and 2G memory. On 10 of them, we deploy our system directly. On the remaining 18 physical machines, we allocate the total of 190 virtual machines and deploy our system on each VM. We compose a client cluster of size s with the 10 physical machines and $(s - 10)$ randomly selected virtual machines. Thus, we can evaluate the behavior of a large cluster – up to 200 hosts – while measuring the actual resource consumption on the 10 physical machines. Our results reflect the average numbers over the 10 physical machines.

In the experiment, the machines form a cluster using a separately deployed seed server. The cluster is formed in a random manner: as nodes join, each new node receives the full set of previously joined peers from the seed server and picks a random seed to send a join message. Once the cluster is formed, every node sends out DNS queries at the rate

computed as follows. We first obtain the maximum request rate within 5 second interval for each client according to the trace. Then we order the clients by these rates and use the average request rate among clients in the 50% percentile (e.g., the most active 50% of all the clients), which is about 7.1 req/s. During the experiment, each client sends out 7.1 DNS queries per second for the domain names taken from the list of unique domain names extracted from the trace. The resource consumption of our system components was found to be negligible – within 1.3% of one core for CPU and about 3% for memory – and barely grows with the cluster size. Given these small overhead levels, we omit the graphs due to space limitations.

C. P2P Lookup Delay

We now turn to the delay overhead in our approach due to the P2P lookup. The testbed setup is the same as in Section V-B. Since all machines in Emulab are on the same LAN, the delay between any pair of hosts is around 0.3ms. Thus, any time a P2P routing hop involves two peers running on virtual machines that happen to reside on the same physical machine, we add 0.3ms to the overlay delay. To focus on how much additional delay occurs due to the P2P nature of our system, we (1) turn off DNS caching, so every lookup leads to the P2P communication and (2) exclude the external delay the DNS queries spend outside the cluster, as this delay would be the same in our and the current systems. Consequently, when the LOOKUP message reaches its target peer, the latter constructs a RESPONSE message with the priori known IP address and sends this message to the source host immediately.

In this experiment, we pick a random cluster of clients of a given size and assign each client to each peer in the testbed. We then extract HTTP accesses of these clients from the trace, and for each access, perform a DNS lookup for the URL's hostname from the corresponding peer. Bamboo DHT offers two options for message routing, "send-back" where response messages are sent back directly from the target to source peer, and "route-back" where response messages are routed from the target to source peer through the overlay network. In our experiments, we consider both options.

Figure 4 shows the results. We can see that the lookup delay is between 2ms and 3.5ms. Furthermore, the delay is virtually flat as the cluster size increases in the case of the route-back, and grows only marginally in the case of send-back (we are puzzled by the higher delay of the send-back option; we can only speculate that it is an implementation artifact of Bamboo). This, along with the fact that the send-back, which incurs fewer P2P routing hops, exhibits higher delay, suggests that the overall delay is dominated by processing at the source and target peers rather than by the routing itself.

Obviously, the delay cost of a P2P routing hop will be higher for some residential clusters, such as those with DSL-connected peers. Yet other popular access technologies, namely cable modems, represent broadcast medium with very low RTT between hosts on the same coaxial cable. Because one cable typically connects hundreds of homes, these neighborhoods would be prime candidates to form a P2P cluster. Further, newer access network technologies such as fiber-to-the-home (FTTH) are gaining rapid adoption and exhibit low RTTs similar to organizational LANs. Finally, note that a host would generate a lookup message only for local cache misses, and according Figure 4, the local hit rate for a stand-alone host is around 60%, meaning that most lookups will not incur any delay. In fact, the shared cache hit ratio grows to 80% for three hosts, suggesting a possibility for hosts in individual home networks to form their own clusters. In summary, the P2P routing delay in our approach is negligible for clusters of hosts with cable modem or FTTH Internet connectivity and acceptable in multi-computer home networks with DSL connectivity. A further study is needed to assess its suitability for clusters of individual DSL-connected hosts.

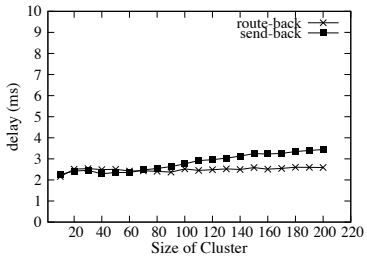
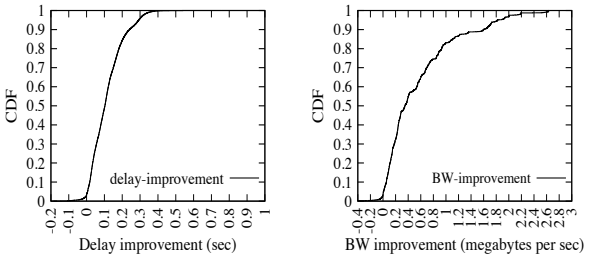


Fig. 4. Routing delay

D. End-to-end Improvement

Finally, we study the improvement in user-perceived service quality our system promises, depending on the distance of the clients from their LDNSs. We investigate this problem by comparing the performance of Akamai CDN servers returned by LDNSs with different network distances to the clients. We consider two metrics here: network delay and the effective bandwidth of a web download (BW). We select 100 PlanetLab nodes distributed around the world as clients and obtain the set of LDNS servers as follows. We first get a list of Gnutella peer IP addresses from [15] and a list of web server IP addresses from [16]. Then we perform reverse DNS queries to get the authoritative DNS server (ADNS) for each of these IPs and filter out those ADNSs that reject external recursive DNS queries as they could not be used in our experiment. As a result, we were able to identify 13420 ADNSs that could act as LDNS for our clients.

In the experiment, each client first obtains the ping distance to all the LDNSs. Then it sends DNS query for the domain name of a URL that is delivered by Akamai to each LDNS and gets the corresponding Akamai server IP address selected by Akamai. After that, the client collects the ping distance to each Akamai server (averaged over 3 tries) and the effective bandwidth of downloading the URL from that server (using the *curl* command, averaged over 5 tries). To ensure every download occurs from the Akamai server's cache and not from the original Web site, we download the object first from each Akamai server (thus bringing the object into the server's cache) before performing the actual measurement.

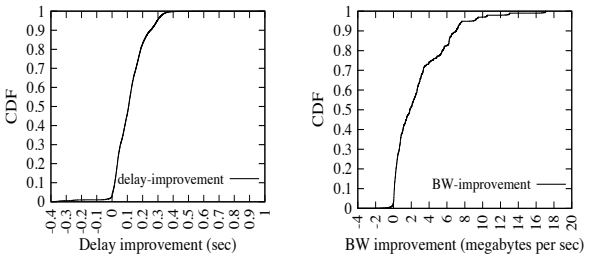


(a) CDF of delay improvement (b) CDF of BW improvement

Fig. 5. End-to-end improvement (object size 6.8k)

Since the effective download bandwidth depends on the object size, our experiment used three Akamai-delivered objects with sizes 6.8k, 54k and 2M. Among 100 planetlab nodes, we were only able to get complete results from 85 of them due to node failures and other unknown errors. We also eliminate the results in which we fail to get the ping distance from clients to LDNS or Akamai servers due to the filtering of ICMP packets.

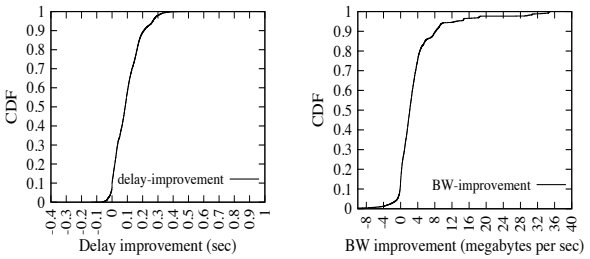
Figures 5 – 7 show CDFs of the reduction in latency and increase in download bandwidth of the Akamai server selected by Akamai when we used the LDNS co-located with the client over Akamai servers selected when using decoupled LDNSs. We can see significant end-to-end improvement by running LDNS on local machine in the overwhelming majority of cases.



(a) CDF of delay improvement (b) CDF of BW improvement

Fig. 6. End-to-end improvement (object size 54k)

To see how the improvements depend on the distance between the client and LDNS, we calculate the Spearman's rank correlation between the distance from a given client to each LDNS and our two quality metrics of the CDN servers selected by Akamai when we used that LDNS. Table I presents



(a) CDF of delay improvement (b) CDF of BW improvement

Fig. 7. End-to-end improvement (object size 2M)

TABLE I
RANKING CORRELATION

Image size	Delay-delay correlation	Delay-BW correlation
6.8k	0.881	-0.869
54k	0.882	-0.861
2M	0.826	-0.729

the average values of the Spearman's correlation coefficient for these two metrics computed over all the clients and all three object sizes. The results show very strong correlation between the client/LDNS distance and both server quality metrics we used. We conclude that bringing LDNS as close as possible to the clients directly improves the quality of server selection by Akamai.

VI. CONCLUSION

In this work, we observed that the distance between clients and their LDNS servers can have a significant negative impact on DNS-based server selection, which is widely used by content deliver networks (CDNs) and other platforms. We then proposed a novel peer-to-peer client-side DNS mechanism that moves LDNS close to their clients while still allowing nearby clients to share the common DNS cache. Through trace-driven simulations and tests on a real prototype setup, we showed that our approach holds significant promise of facilitating better server selection by CDNs. In the future, we plan to investigate security issues of our scheme, in particular, making it resilient to situations when some clients in a cluster are compromised.

REFERENCES

[1] A. Barbir, B. Cain, R. Nair, and O. Spatscheck, "Known Content Network (CN) Request-Routing Mechanisms," *IETF RFC 3568*, 2003.
[2] Z. M. Mao, C. D. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, and J. Wang, "A precise and efficient evaluation of the proximity between Web clients and their local DNS servers," *USENIX ATC*, 2002.
[3] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of DNS-based server selection," *IEEE INFOCOM*, 2001.
[4] B. Krishnamurthy and J. Wang, "On network-aware clustering of web clients," *ACM SIGCOMM*, 2000.
[5] R. Cox, A. Muthitacharoen, and R. T. Morris, "Serving DNS using a peer-to-peer lookup service," *IPTPS*, 2002.
[6] V. Ramasubramanian and E. G. Sirer, "The design and implementation of a next generation name service for the internet," *ACM SIGCOMM*, 2004.
[7] K. Park, V. Pai, L. Peterson, and Z. Wang, "CoDNS: Improving DNS performance and reliability via cooperative lookups," *USENIX OSDI*, 2004.
[8] "http://www.maxmind.com/app/ip-location,"
[9] C. Huang, A. Wang, J. Li, and K. W. Ross, "Measuring and evaluating large-scale CDNs," *ACM SIGCOMM Internet Measurement Conf. (paper withdrawn)*, 2008.
[10] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS performance and the effectiveness of caching," *IEEE/ACM Transactions on Networking*, vol. 10, pp. 589–603, 2002.
[11] "http://bamboo-dht.org,"
[12] H. Ballani and P. Francis, "Mitigating DNS DoS attacks," *ACM CCS*, 2008.
[13] "http://www.bind9.net,"
[14] "http://www.emulab.net,"
[15] "http://mirage.cs.uoregon.edu/p2p/snapshots.html,"
[16] "http://www.icir.org/tbit/urllistfeb2004.txt,"