# The Anatomy of LDNS Clusters: Findings and Implications for Web Content Delivery

Hussein A. Alzoubi
EECS Department
Case Western Reserve Univ.

Michael Rabinovich
EECS Department
Case Western Reserve Univ.

Oliver Spatscheck
AT&T Research Labs

## ABSTRACT

We present a large-scale measurement of clusters of hosts sharing the same local DNS servers. We analyze properties of these "LDNS clusters" from the perspective of content delivery networks, which commonly use DNS for load distribution. We found that not only LDNS clusters differ widely in terms of their size and geographical compactness but that the largest clusters are actually extremely compact. This suggests potential benefits of a load distribution strategy with nuanced treatment of different LDNS clusters based on the combination of their size and compactness. We further observed interesting variations in LDNS setups including a wide use of "LDNS pools" (which as we explain in the paper are different from setups where end-hosts simply utilize multiple resolvers).

## Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; C.4 [ **Performance of Systems**]: Reliability, availability, and serviceability

## General Terms

Measurement, Performance

## Keywords

DNS, Request Routing, Content Delivery Networks

## 1. INTRODUCTION

Domain Name System (DNS) is a key component of the today's Internet apparatus. Its primary goal is to resolve human-readable host names, such as "cnn.com" to hosts IP addresses. In particular, HTTP clients send queries for these resolutions to their *local DNS servers* (LDNS), which route these queries through the DNS infrastructure and ultimately send them to *authoritative DNS servers* (ADNS) that maintain the needed mapping information. The ADNS servers then return the corresponding IP addresses back to LDNS, which forward them to the clients, who then can proceed with their HTTP interactions. By returning different IP addresses to different queries, ADNS can direct different HTTP requests to different servers. This commonly forms the basis for transparent client request routing in replicated web sites, content delivery networks (CDNs), and – more recently – cloud computing platforms.

When selecting an IP address to reply to a DNS query, ADNS only know the identity of the requesting LDNS and not the client that originated the query. Thus, the LDNS acts as the proxy for all its clients. We call the group of clients "hiding" behind a common LDNS server an "LDNS cluster". DNS-based network control can only distribute client demand among data centers or servers at the granularity of the entire LDNS clusters, leading to two fundamental problems, *hidden load problem* [6], which is that a single load balancing decision may lead to unforeseen amount of load shift, and the *originator problem* [21], which is that when the request routing apparatus attempts to route clients to the nearest data center, the apparatus only consider the location of the LDNS and not the clients behind.

This paper studies properties of LDNS clusters from the perspective of their effect on client request routing. Various proposals were put forward to include the identity of the client into the LDNS queries but they have not been adopted, presumably because these proposals are incompatible with shared LDNS caching, where the same response can be reused by multiple clients. There is another such proposal currently underway, spearheaded by Google [8]. Our work will be useful in informing these efforts.

The key findings in our study are the following:

- An overwhelming majority of the clusters, even as seen from a very high-volume web site, are very small, posing no issue with respect to hidden load. However, despite recent trends transforming busy resolvers into complex distributed infrastructures (e.g. anycast based platforms such as [17, 9]) there remain a few "elephant" clusters [1]). Thus, a DNS-based request routing system may benefit by tracking the elephant LDNS clusters and treating them differently.

- LDNS clusters differ widely in terms of their geographical and autonomous system (AS) span. Furthermore, the extent of this span does not correlate with cluster size: the busiest clusters are very compact geographically but not in terms of AS-sharing. Thus, a DNS-based request routing system can benefit by treating LDNS clusters differently depending on a combination of their size and compactness: when there is a need

[1]Note that our measurement setup is able to distinguish clients behind individual resolver nodes in these platforms as distinct clusters, so we do not conflate these platforms into an elephant cluster.


*WWW 2013*, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

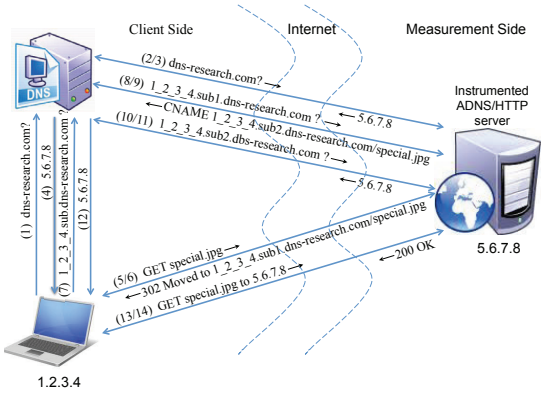


**Figure 1: Measurement Setup.**

to rebalance server load, the system may re-route requests from non-compact clusters first because they benefit less from proximity-sensitive routing anyway.

- A large number of IPs act as both Web clients and their own LDNSs. We find evidence that much of this phenomenon is explained by the presence of middleboxes (NATs, firewalls, and web proxies). However, although they aggregate traffic from multiple hosts, these clusters exhibit, if anything, lower activity. Hence this aspect by itself does not appear to warrant special treatment from the request routing system.

- We find strong evidence of LDNS pools with shared cache, where a set of "worker servers" shares work for resolving clients' queries. While the implications of this behavior for network control remain unclear, the prevalence of this LDNS behavior warrants a careful future study.

We stress that our characterization is done from the vantage point of a busy web site, i.e., based on the activity seen by this site. For instance, when we consider an LDNS cluster size, this size reflects the clients that visited our Web site over the duration of our study. There may be other hosts behind this LDNS that we would not have seen. Our vantage point, however, is an example of what a busy web site may face when performing request routing.

## 2. SYSTEM INSTRUMENTATION

To characterize LDNS clusters (i.e., sets of hosts behind a given LDNS), we need to associate hosts with their LDNSs. We used an enhanced approach from our prior work [15] to gather our measurements. As shown in Figure 1, we deploy a measurement machine that runs both a custom authoritative DNS server for a domain we registered for the purpose of this experiment (dns-research.com) and a custom Web server. The Web server hosts a special image URL, `dns-research.com/special.jpg`.

When a user accesses this URL, the following steps occur:

- The user's browser sends a DNS query to its local DNS server to resolve dns-research.com. We call dns-research.com a "base domain" and a query for it "base query" (step 1 in the figure).

- The LDNS recursively resolves this query, ultimately our DNS server (steps 2 and 3) and returns the result (the IP address of our measurement machine) to the client (step 4).

- The client sends the HTTP request for special.jpg to our Web server (step 5). Our server responds with an HTTP redirect ("302 Moved") specifying another URL in the `dns-research.com` domain (step 6). Our server constructs this new URL dynamically by embedding the client's IP address into the hostname of the URL. For example, when our Web server receives a request for *special.jpg* from client 206.196.164.138, the Web server replies to the client with the following redirection link: `206_196_164_138.sub1.dns-research.com/special.jpg`.

- Following the redirection, the client issues another DNS request to its LDNS - for hostname that embeds its own IP address, in the example above (step 7) the request URL is `206_196_164_138.sub1.dns-research.com`. The LDNS eventually sends this request to our DNS server (step 8), which can now record both the IP address of the LDNS that sent the query and the IP address of its associated client that had been embedded in the hostname. Thus, the association of the client and its LDNS is accomplished.

In the original approach of [15], ADNS server would now complete the interaction by resolving the query to the IP of our Web server, which would respond to the client's HTTP request with a 1-pixel image. We augmented this approach as follows. Our ADNS responds to a query for **.sub1.dns-research.com* with the corresponding CNAME **.sub2.dns-research.com*, where * denotes the same string representing client's IP address (step 9), forcing the client to perform another DNS resolution for the latter name. Moreover, our DNS server includes its own IP address in the authority section of its reply to the "sub1" query, which ensures that the LDNS sends the second request ("sub2" request) directly to our DNS server. We added this functionality to discover *LDNS pools* (Section 8.2). Upon receiving the "sub2" query (step 10), our ADNS returns its own IP address (steps 11-12), which is also the IP address of our Web server, and the client performs the final HTTP download of our special image (steps 13-14).

We have partnered with a high-volume consumer-oriented Web site[2], which embedded the base URL for our special image into their home page. This allowed us to collect a large amount of measurement data as discussed below. To obtain repeated measurements from a given client,, we used a low 10 seconds TTL for our DNS records – lower than any CDN we are aware of – and added a "cache-control:no-cache" HTTP header field to our HTTP responses.

## 3. THE DATASET

The measurement data included the DNS and HTTP logs collected at our measurement host. The DNS logs contained the timestamp of the query, the IP address of the requesting LDNS, query type, and query string, and the HTTP logs

[2]Part of the conditions for this collaboration is that we are unable to name the site.

**Table 1: High-level dataset characterization**

| | |
|---|---|
| Unique LDNS IPs | 278,559 |
| Unique Client IPs | 11,378,020 |
| Unique Client/LDNS IP Pairs | 21,432,181 |

contained the request time and User-Agent and Host headers. We conducted our measurements over 28 days, from Jan 5th, 2011 to Feb 1st. During this period, we collected the total of over 67.7 million sub1 and sub2 DNS requests and around 56 million of the HTTP requests for the final image (steps 13/14 in Figure 1; we refer to these final HTTP requests as simply HTTP requests in the rest of the paper, but stress that we do not include the initial redirected HTTP requests in steps 5-6 of the setup into any of the results). The higher number of HTTP requests compared to DNS queries (indeed, as Figure 1 shows, a client access should generate a sub1 and sub2 DNS request for a final HTTP request) is due to the well known fact that clients and LDNSs reuse DNS responses much longer than the TTL values assigned by the ADNS [18]. We verified that some HTTP accesses occur long past the 10s (our TTL) since the preceding sub1 and sub2 queries.

Table 1 shows the overall statistics of our dataset. Our measurements include over 11.3M clients and almost 280K LDNS resolvers representing, respectively, 17,778 and 14,627 autonomous systems (ASs). We have obtained over 21M unique *associations* between these clients and LDNSs, where an association (or *pair*) connects a client and the LDNS used by this client for a DNS resolution.

We refer to all clients that used a given LDNS as the *LDNS cluster*. Thus, an LDNS cluster contains one LDNS IP and all clients that used that LDNS in our experiment. Note that the same client can belong to multiple LDNS clusters if it used more than one LDNS during our experiment.

# 4. CLUSTER SIZE

We begin by characterizing LDNS clusters in terms of their size. This is important to DNS-based server selection because of the *hidden load problem* [6]: a single DNS response to an LDNS will direct HTTP load to the selected server from all clients behind this LDNS for the TTL duration. Uneven hidden loads may lead to unexpected results from the load balancing perspective. On the other hand, knowing activity characteristics of different clusters would allow one to take hidden loads into account during server selection process. For example, dynamic adjustments of the TTL in DNS responses to different LDNSs can be used to compensate for different hidden loads [5, 6].

We characterize LDNS cluster sizes from two perspectives - the number of clients behind a given LDNS and the amount of activity originated from all clients in the cluster. We should stress that the former is done purely based on IP addresses, and our use of the term "client" is simply a shorthand for "client IP address". It has been shown that IP addresses may not be a good representation of individual hosts due to the presence of network address translation boxes and dynamic IP addresses [14, 4]. We characterize cluster sizes from the perspectives of the number of clients in a cluster and the amount of access activity originated from a cluster.

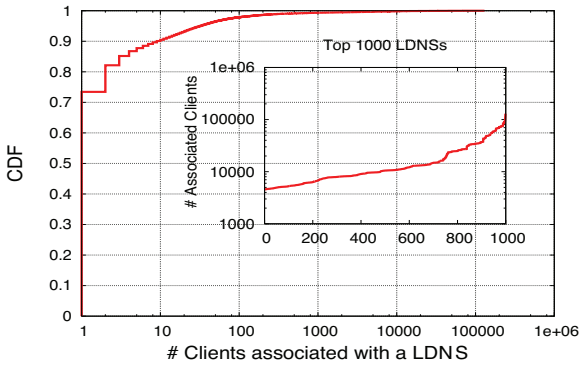


**Figure 2: Distribution of LDNS cluster sizes.**

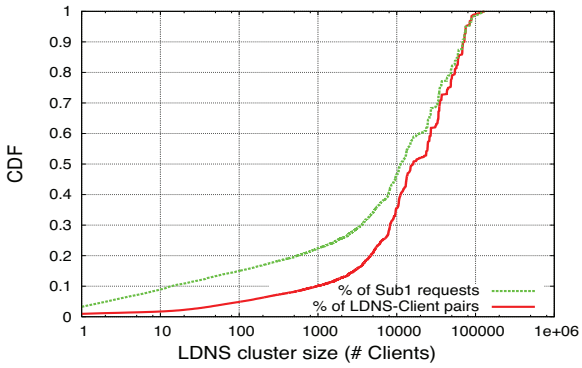


**Figure 3: Distribution of sub1 requests and client/LDNS pairs attributed to LDNS clusters of different sizes**

## 4.1 Number of Clients

Figure 2 shows the CDF of LDNS cluster sizes while the cut-in subfigure shows the sizes of the 1000 largest LDNS clusters (in the increasing order of size). We found that a vast majority of LDNS clusters are small - over 90% of LDNS clusters contain fewer than 10 clients. This means that most *clusters* do not provide much benefit of shared DNS cache to their clients when they access our partner Web site. To see the potential impact on *clients*, Figure 3 shows the cumulative percentage of sub1 requests issued by LDNSs representing clusters of different sizes as well as cumulative percentages of their client/LDNS associations. More precisely, for a given cluster size $X$, the corresponding points on the curves show the percentage of sub1 requests issued by LDNS clusters of size up to $X$, and the percentage of all client/LDNS associations belonging to these clusters. As seen on the figure, small clusters, with less than 10 clients, only contribute less than 10% of all sub1 requests and comprise less than 1% of all client/LDNS associations. Thus, even though these small clusters represent over 90% of all LDNS clusters, an overwhelming majority of clients belong to larger clusters, which are also responsible for most of the activity. THus, most clients are not affected by limited shared DNS caching in small clusters.

Moreover, despite the prevalence of DHCP-driven DNS configuration of end-hosts and – more recently – anycasted resolvers, both facilitating distributed resolver infrastructures, we still observed a few "elephant" clusters. The largest

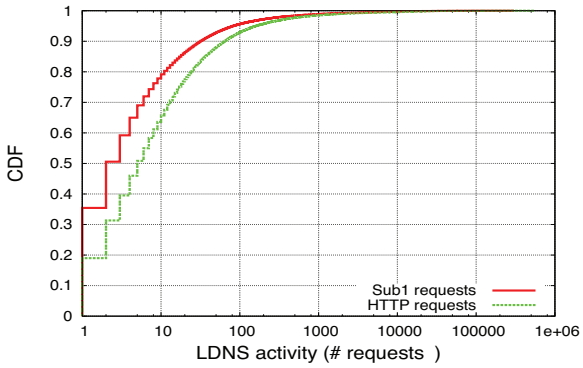


**Figure 4: LDNSs Activity in terms of DNS and HTTP requests.**

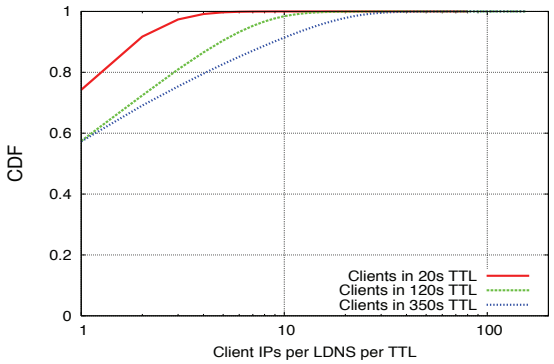


**Figure 5: LDNS cluster sizes within TTL windows (all windows).**

cluster (with LDNS IP 167.206.254.14) comprised 129,720 clients and it alone was responsible for almost 1% of all sub1 requests. Elephant clusters may affect dramatically load distribution, and their small number suggests that it might be warranted and feasible to identify and handle them separately from the rest of the LDNS population. Overall, the size of LDNS clusters ranged from 1 to 129,720 clients, with the average size being 76.94 clients. We further consider top-10 elephant LDNS clusters in Section 7.

## 4.2 Cluster Activity

We now turn to characterizing the activity of LDNS clusters. We characterize it by the number of their sub1 requests as well as by the number of the final HTTP requests. Since a client may belong to multiple LDNS clusters (e.g., when it used a different LDNS at different times), we associate an HTTP request with the last LDNS that was used by the client prior to the HTTP request in question.

Figure 4 shows the CDF of the number of sub1 queries issued by LDNSs, as well as the CDF of the number of HTTP requests issued by clients behind each LDNS during our experiment. Again, both curves in the figure indicate that there are only a small number of high-activity clusters. Indeed, 35% of LDNSs issued only one sub1 request, and 96% of all LDNSs issued less than 100 sub1 requests over the entire experiment duration. Yet the most active LDNS sent 303,042 sub1 requests. The HTTP activity presents similar trends although we do observe some hidden load effects even among low-activity clusters: whereas 35% of LDNSs issued a single DNS query, only less than 20% of their clusters issued a single HTTP request. This is due to DNS caching, which often extends beyond our low TTL of 10s.

Overall, our observations of LDNS cluster sizes, both from the number of clients and activity perspectives, confirm that platforms using DNS-based server selection may benefit from treating different LDNSs differently.

At the same time, they may only need to concentrate on a relatively small number of "elephant" LDNSs for such special treatment.

# 5. TTL EFFECTS

The above analysis considered the LDNS cluster activity over the duration of the experiment. However, platforms that use DNS-based server selection, such as CDNs, usually assign relatively small TTL to their DNS responses to retain an opportunity for further network control. In this section, we investigate the hidden loads of LDNS clusters observed within typical TTL windows utilized by CDNs, specifically 20s (used by Akamai), 120s (AT&T's ICDS content delivery network) and 350s (Limelight).

In order to get the above numbers, we use our DNS and HTTP traces to emulate the clients' activity under a given TTL. The starting idea behind this simulation is simple: the initial sub1 query from an LDNS starts a TTL window, and then all subsequent HTTP activity associated with this LDNS (using the procedure described in Section 4.2 ) is "charged" to this window; the next sub1 request beyond the current window starts a new window. However, two subtle points complicate this procedure.

First, if after the initial sub1 query to one LDNS, the same client sends another DNS query through a different LDNS within the emulated TTL window (which can happen since the actual TTL in our experiments was only 10s) we "charge" these subsequent queries and their associated HTTP activity to the TTL window of the first LDNS. This is because with the longer TTL, these subsequent queries would not have occurred since the client would have reused the initial DNS response from its cache.

Second, confirming the phenomenon previously measured in [18], we have encountered a considerable number of requests that violated TTL values, with violations sometimes exceeding the largest TTL values we simulated (350s). Consequently, in reporting the hidden loads per TTL, we use two lines for each TTL value. The lines labeled "strict" reflect only the HTTP requests that actually fell into the TTL window[3] Thus, these results ignore requests that violate the TTL value. The "non-strict" lines include these violating HTTP requests and count them towards the hidden load of the TTL window to which the associated DNS query was assigned.

Figure 5 shows the CDF of the LDNS cluster sizes observed for each LDNS in each TTL window, i.e., each LDNS contributed a separate data point to the CDF for each window (the full paper [2] also considers average cluster sizes ob-

[3]For simplicity of implementation with also counted HTTP requests whose corresponding sub* queries were within the window but the HTTP requests themselves were within our real TTL of 10s past the window. There were very small number of such requests (a few thousand out of 56M total) thus this does not materially affect our results.

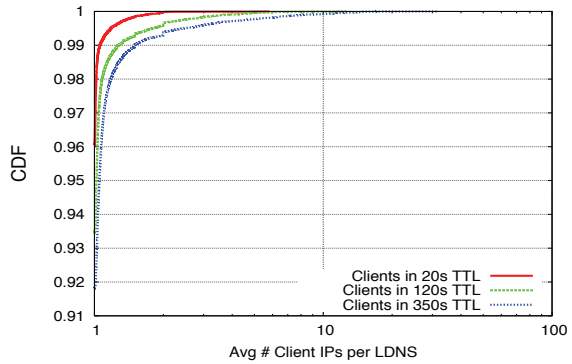


**Figure 6: Average LDNS cluster sizes within a TTL window (averaged over all windows for a given LDNS)**

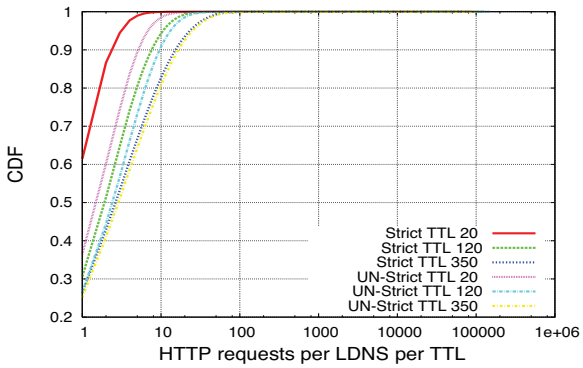


**Figure 7: HTTP requests within TTL windows (all windows).**

served in all TTL windows for given clusters). The majority of windows, across all LDNSs, contained only one client. As expected, in larger the TTL windows, the number of clients an LDNS serves increases. Still, only around 10% of TTL intervals had more than 10 clients under TTL of 350s, and less than 2% of the intervals had more than 10 clients with TTL of 120s.

Figure 6 shows the CFP of the average in-TTL cluster sizes for LDNSs across all their TTL intervals. That is, each LDNS contributes only one data point to the CDF, reflecting its average cluster size for all its TTL intervals[4]. The average in-TTL cluster sizes are even smaller, with virtually all LDNSs exhibiting average in-TTL cluster size below 10 clients under all TTL values. The difference between the two figures is explained by the fact that busier LDNSs (i.e., those showing up with more clients within a TTL) tend to appear more frequently in the trace, thus contributing more data points in Figure 5.

To assess how the hidden load of LDNSs depends on TTL, Figures 7 and 8 show, respectively, the CDFs of the number of HTTP requests in all TTL windows and average in-TTL number of HTTP requests for all LDNS across all their TTL intervals. A few observations are noteworthy. First, the difference between strict and non-string lines in Figure 7 indicate violations of the TTL we considered; as expected, these violations decrease for larger TTL and, importantly, all but disappear for TTL of 350 sec. This shows that at these TTL levels, a CDN might not need to be concerned about unforeseen affect of these violations on hidden load. Second, while there re sizable differences in hidden loads among some LDNSs for some TTL values are significant, their absolute values are small overall - virtually all windows contain fewer than 100 requests even for the largest TTL of 350s (Figure 7). Thus, low TTL values are important not for proper load-balancing granularity in routine operations but mostly to react quickly to unforeseen flash crowds. It is obviously undesirable to have to pay overhead on routine operation while using it only for extraordinary scenarios. A better knob would be desirable and should be given consideration in future Internet architectures.

[4]The average in-TTL cluster sizes per-LDNS may reflect better the kind of input data available to a request routing algorithm.

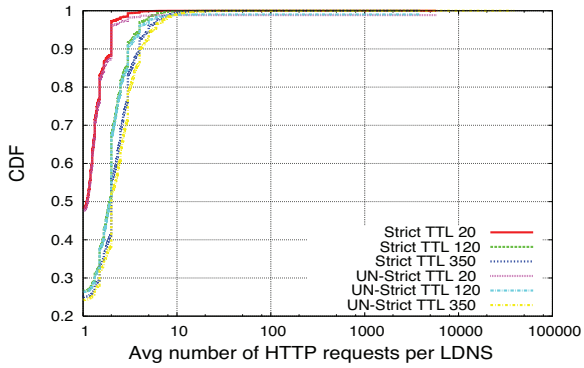


**Figure 8: Average number of HTTP requests per LDNS within a TTL window (averaged over all windows for a given LDNS).**

## 6. CLIENT-TO-LDNS PROXIMITY

We consider the proximity of clients to their LDNS servers, which determines the severity of the *originator problem* and can have other implications for proximity-based request routing. Prior studies [23, 15] looked at several proximity metrics, including TCP traceroute divergence, network delay difference as seen from a given external vantage point, and autonomous system sharing - how many clients reside in the same AS as their LDNS servers. We revisit the AS-sharing metric, but instead of the other metrics, which are vantage-point dependent, we consider the air-mile distance between clients and their LDNSs. Ideally we would also have liked to know the network delay between these parties but we have no visibility into this metric from our vantage point.

### 6.1 Air-Miles Between Client and LDNS

We utilized the GeoIP city database from Maxmind [16], which provides the geographic location information for IP addresses, to study geographical properties of LDNS clusters. Using the database dated from February 1, 2011 (so that our analysis would reflect the GeoIP map at the time of experiment), we mapped the IP addresses of the clients and their associated LDNSs and calculated the geographical distance ("air-miles") between them.

Figure 9 shows the cumulative distribution function (CDF) of air-miles of all client/LDNS pairs. The figure shows that

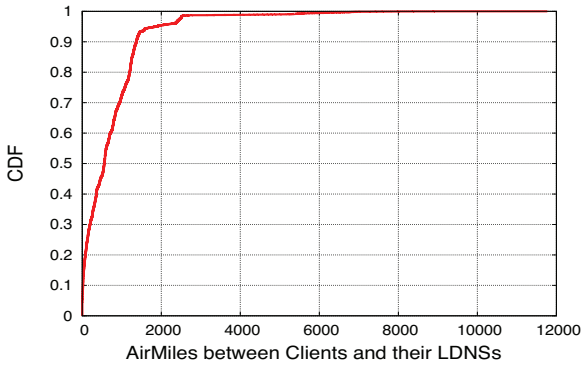


**Figure 9: Air miles for all client/LDNS pairs**

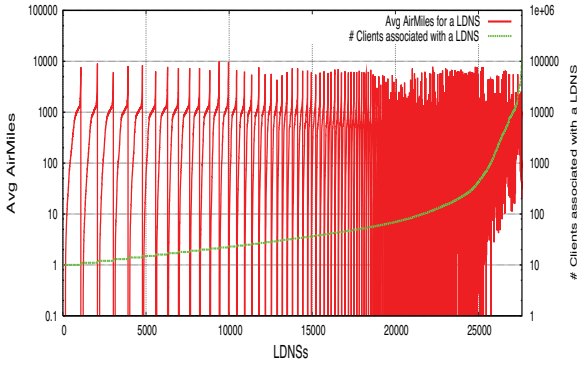


**Figure 10: Avg client/LDNS distance in top LDNS clusters**

clients are sometimes situated surprisingly far from their LDNS servers. Only around 25% of all client/LDNS pairs were less than 100 miles apart while 30% were over 1000 miles apart. This suggests an inherent limitation to how accurate, in terms of proximity, DNS-based server selection can be. We note that our measurements show significantly greater distances than previously measured in [11] (see Section 10 for more details).

## 6.2 Geographical Span

We are also interested in the geographical span of LDNS clusters. Geographically compact clusters are more amenable to proximity routing than the spread-out ones. If a content platform can distinguish between these kinds of clusters, it could treat them differently: requests from an LDNS representing concentrated cluster could be preferentially resolved to a highly proximal content server, while requests from LDNSs representing spread-out clusters could be used to even out load with less regard for proximity. This would result in more requests resolving to proximal content servers when it actually counts.

For space consideration, we focus on the LDNSs with more than 10 clients, which represent almost 10% of all LDNSs in the data set (see the full paper for the results on small clusters [2]). Figure 10 plots, for each such LDNS server, the average airmiles from this server to all its clients and the number of clients for the same LDNS. The X-axis shows LDNSs sorted by the size of their client cluster, and within LDNSs of equal cluster size, by the average air miles distance. shows the average air-miles and the number of clients for these top LDNS clusters. The "teeth" in the graph are due to the above sorting of LDNSs: each "tooth" represents a set of all LDNS clusters with the same number of clients, and the tooth-like shape reflects the fact that each such set, except for the sets comprising the largest clusters, contains clusters with the average geographical span ranging from 0 to up to 10,000 miles.

As the number of clients increases, the variation of geographical span among clusters narrows but still remains significant, with an order of magnitude differences between clusters. This provides evidence in support of differential treatment of LDNSs not just with respect to their differences in size and activity as we saw in Sections 4 and 4.2 but also with respect to proximity-based server selection.

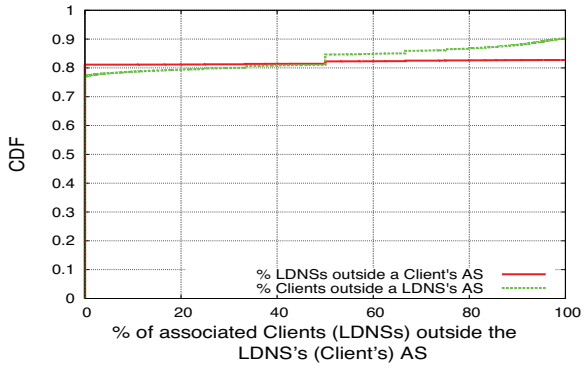


**Figure 11: CDF of LDNS clusters with a given % of clients/LDNSs outside their LDNS's/Client's autonomous system.**

**Table 2: Client activity attributed to client-LDNS associations sharing the same AS**

| DNS requests | HTTP requests |
|---|---|
| 73.79% | 81.97% |

## 6.3 AS Sharing

Another measure of proximity is the degree of AS sharing between clients and their LDNSs.

Figure 11 shows this information from, respectively, LDNS and client perspective. The LDNS perspective reflects, for a given LDNS, the percentage of its associated clients that are in the same AS as the LDNS itself. The clients' perspective considers, for a given client, the percentage of its associated LDNSs that are in the same AS as the client itself.

While almost 77% of LDNSs have all their clients in the same AS as they are, 15% of LDNSs have over half of their clients outside their AS and 10% have *all* their clients in a different AS. From the clients' perspective, we found that more than 9 million client have all the LDNSs in their AS while nearly 2 million (almost 17%) have *all* their LDNSs in a different AS. Only a small number of clients - over 180K had a mix of some LDNSs within and some LDNSs outside the client's AS. Such a strong dichotomy (i.e., that clients either had all or none of their LDNSs in their own AS) is explained by the fact that most clients associate with only a small number of LDNSs.

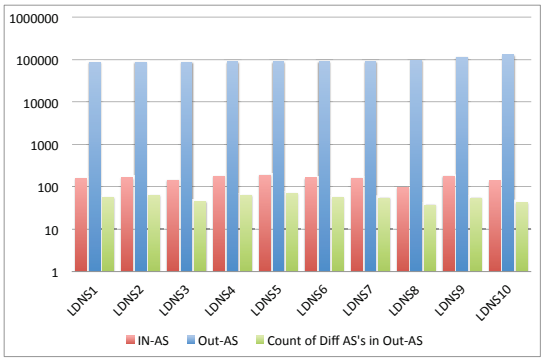


**Figure 12: AS sharing of top-10 LDNSs and their clients**

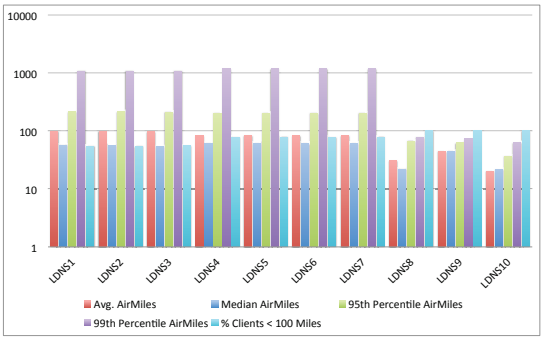


**Figure 13: Air miles between top-10 LDNSs and their clients.**

Our discussion so far concerned the prevalence of AS sharing in terms of client population. In other words, each client-LDNS association is counted once in those statistics. However, different clients may have different activity levels, and we now consider the prevalence of AS sharing from the perspective of clients' accesses to the Web site. Table 2 shows the fraction of client activity stemming from client-LDNS associations that share the same AS. The first column reports the fraction of all sub1 and sub2 request pairs with both the client and LDNS belonging to the same AS. This metric reflects definitive information but it only approximates the level of client activity because of the 10s TTL we use for sub1 and sub2 responses: we do not expect the same client to issue another DNS query for 10 seconds (or longer, if the client violates TTLs) no matter how many HTTP requests it issues within this time. The second column shows the fraction of all HTTP requests such that the preceding DNS query that originated from the same client used an LDNS in the same AS as the client. This metric reflects definitive activity levels but is not iron-clad in attributing the activity to a given client/LDNS association.

First, we note that the prevalence of AS sharing measured based on activity is somewhat lower than based on client populations.

Second, these levels of AS sharing are still significantly higher than those reported in the 10-year old study [15] (see Table 5 there). This is an encouraging development for DNS-based request distribution.

Overall, while the prevalence of AS sharing increased form 10 years ago, we found a sizable fraction (15 - 17%) of client/LDNS associations where clients and LDNSs reside in different ASs. One of the goals in server selection by CDNs, especially those with a large number of locations such as Akamai, is to find an edge server sharing the AS with the originator of the request [20]. Our data shows fundamental limits to the benefits from this approach.

# 7. TOP-10 LDNS CLUSTERS

We have investigated the top 10 LDNSs manually through reverse DNS lookups, namely whois records, and MaxMind ISP records for their IP addresses. The top-10 LDNSs in fact all belong to just two ISPs, which we refer to as ISP1 (LDNSs ranked 10-4), and ISP2 (ranked 3-1). The top three clusters of ISP2 contributed 1.6% of all unique client-LDNS associations in our traces and 2.33% of all sub1 requests. The extent of the AS sharing for these clusters is shown in Figure 12. In the figure, the bars for each cluster represent (from left to right) the number of clients sharing the AS with the cluster's LDNS, the number of clients in other ASs, and for the latter clients, the number of different ASs they represent. The figure shows very low degree of AS sharing in the top clusters. Virtually all clients belong to a different AS from the one where their LDNS resides, and each cluster spans dozens of different ASs. We further verified that these ASs belong to different organizations from those owning the corresponding LDNSs. Interestingly, the AS sharing is very similar between ISP1 and ISP2.

We also consider the geographical span of the top 10 clusters using MaxMind GeoIP city database. Figure 13 shows the average and median air-miles distance between the LDNS and its clients, as well as the 95th and 99th percentiles for these distances for each LDNS Cluster. The last bar shows the percentage of clients in that cluster that are less than 100 Mile apart from the LDNS.

While figure 12 shows that top-10 LDNS clusters spans dozens of ASs which suggests topologically distant pairs, Figure 13 shows that these clusters are very compact, with most clients less than 100 miles away from their LDNSs. Further, although both ISPs exhibit similar trends, ISP2 clearly has more ubiquitous LDNS infrastructure and more of their customers can expect better service from CDN-accelerated Web sites. Indeed, ISP2's LDNSs have more than 99.9% of their clients within 100 AirMiles radius, with the average range between 20 - 43 AirMiles.

# 8. CLIENT SITE CONFIGURATIONS

We now discuss some noteworthy client and LDNS behaviors we observed in our experiments.

## 8.1 Clients OR LDNSs?!

Our first observation is a wide-spread sharing of DNS and HTTP behavior among clients. Out of 278,559 LDNS servers in our trace, 170,137 or 61.08% also show up among the HTTP clients. We refer to these LDNSs as the "Act-Like-Clients" group.

A vast majority of these LDNSs – 98% or 166,859 – have themselves among *their own* associated clients. We will call these LDNSs the Self-Served group. The other 3278 LDNS IP addresses always used different LDNSs when acted as HTTP clients. Within the Self-Served group, we found that 149,013 of these LDNSs, or 53% of all the LDNSs in our

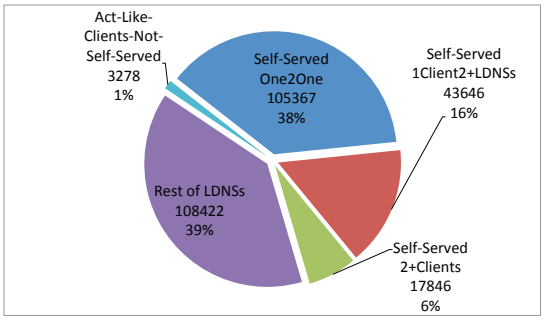


**Figure 14: Distribution of LDNS types**

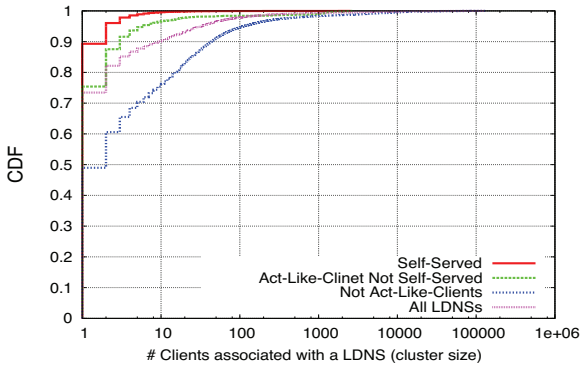


**Figure 15: Cluster size distribution of LDNS groups.**

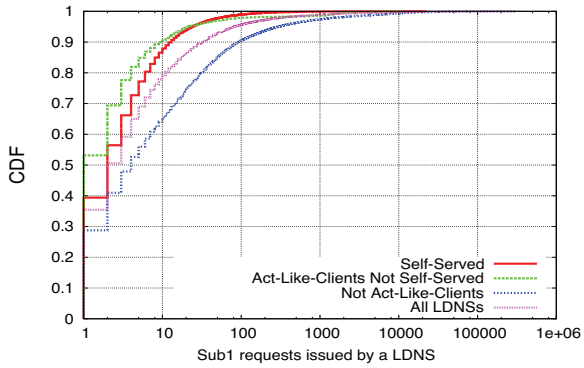


**Figure 16: The number of sub1 requests issued by LDNSs of different types.**

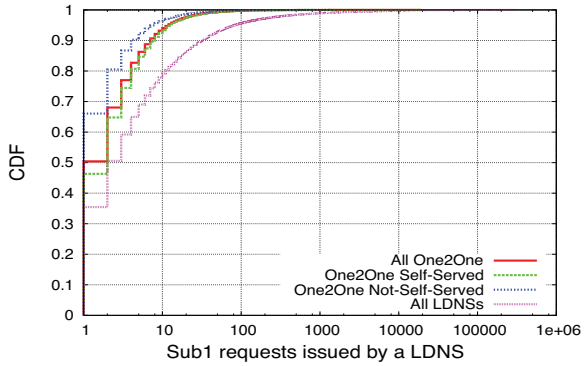


**Figure 17: Number of sub1 requests issued by One2One LDNSs.**

dataset, had themselves as their *only* client during our experiment ("self-served-one-client") while the remaining 17,846 LDNSs had other clients as well ("Self-Served-2+Clients"). Moreover, 105,367 of the self-served-one-client client/LDNS IP addresses never used any other LDNS. We call them the "Self-Served-One2One" group. This leaves us with 43,646 LDNSs that had themselves as their only client but in their client role, they also utilize other LDNSs. This group will be called the "Self-Served-1Client2+LDNSs". Figure 14 summarizes the distribution of these types of LDNSs.

While a likely explanation for the Act-Like-Client Not-Self-Served group is the reuse of dynamic IP addresses (so that the same IP address is assigned to an HTTP client at some point and to an LDNS host at another time), the Self-Served behavior could be caused by sharing of a common middle-box between the LDNS and its clients. In particular, the following two cases are plausible.

- Both clients and their LDNS are behind a NAT or firewall, which exposes a common IP address to the public Internet. A particular case of this configuration is when a home network configures its wireless router to behave as LDNSs. Such configuration is easily enabled on popular wireless routers (e.g., Linksys), although these routers often resolve their DNS queries through ISP LDNS servers [22].

- Clients are behind a proxy that acts as both HTTP proxy/cache and its own LDNS resolver.

We find support for the above explanation using an approach similar to [14]. We utilized the User-Agent headers to identify hosts sharing the same middle-box based on the operating system and browser footprints. We consider an IP address as a possible middle-box if it shows two or more operating systems or operating system versions, or three or more different browsers or browser versions. Out of the total 11.7M clients, we have flagged only 686,651 (5.87%) clients who fall into the above category.[5] However, 51,864 clients among them were from the Self-Served LDNS group, out of the total of 166K such LDNSs. Thus, the multi-host behavior is much more prevalent among self-serving LDNSs than the general client population even though our technique misses single-host NAT'ed networks (which constitute a majority of NAT networks according to [4] although not according to [14]) and NATs whose all hosts have the same platform.

An important question from a CDN perspective is whether these configurations deviate from the "regular" LDNS cluster behavior, in which case they might need special treatment in DNS-based demand distribution. For example, a proxy acting as its own LDNS might show as a small single-client cluster yet impose incommensurately high load on a CDN node as a result of a single act of the CDN server selection.

Figure 15 compares the cluster sizes of self-served and other LDNSs. It shows that self-served LDNS clusters are much smaller than other clusters, in fact they overwhelm-

[5]Compared to previous studies of NAT usage, notably [14] and [4], this finding is more in line with the latter. Note that our vantage point - from the perspective of a Web site - is also closer to [4] than [14].

ingly contain only one client IP address. This finding is in conformance with the behavior expected from a middle-box fronted network. A more revealing finding is displayed in Figure 16, which compares the activity (in terms of sub1 requests) of the self-served LDNSs with other groups.

The figure shows that the self-served LDNS clusters exhibit *lower* activity levels than the not-act-like-clients clusters. Thus, while middleboxes aggregate demand from several hosts behind a single IP address, these middleboxes seem to predominantly front small networks - smaller than other LDNS clusters.

To confirm the presence of demand aggregation in self-served LDNS clusters, Figure 17 factors out the difference in client sizes and compares the activity of Self-Served and Not-Self-Served LDNSs only for One2One clusters.There were 105,367 LDNSs/clients in the One2One Self-Served group and 27,640 in the One2One Not-Self-Served group. Figure 17 shows that the One2One self-served LDNSs in general are indeed more active than Not-Self-Served LDNSs. For instance, 66% of Not-Self-Served LDNSs issued a single request is vs. only 46% of the self-served ones. This increased activity of self-served LDNSs is consistent with moderate aggregation of hosts behind a middle-box.

In summary, we found a large number of LDNSs operating from within middle-box fronted networks - they are either behind the middleboxes or operated by the middleboxes themselves. However, while these LDNSs exhibit distinct demand aggregation, their clusters are if anything less active than other clusters. Thus, a middle-box fronted LDNS in itself does not seem to be an indication for separate treatment in DNS-based request routing.

## 8.2 LDNS Pools

We now consider another interesting behavior. As a reminder, our sub* DNS interactions start with a sub1 request issued by the LDNS to our setup, to which we reply with a sub2 CNAME, forcing the LDNS to send another query, this time for sub2. However, we observed occurrences in our traces where these two consecutive queries (which we can attribute to the same interaction because both embed the same client IP address) came from *different* LDNS servers. In other words, even though we sent our CNAME response to one LDNS, we got the subsequent sub2 query from a different LDNS. Note that this phenomenon is distinct from resolver clusters mentioned in [12]. Indeed those other sets of resolvers occur when clients (ISPs on their behalf) load-balance their original DNS queries among multiple resolvers – the measurements mentioned do not consider which resolvers might handle CNAME redirections. In contrast, in the behavior discussed here, CNAME redirections arrive from different IP addresses.

Such behavior could be caused by an LDNS server with multiple ethernet ports (in which case a server might select different ports for different queries), or by a load-balancing LDNS server farm with shared state. An example of such configuration, hinted by Google in [9], is shown in Figure 18, where two distinct layers of LDNS clusters face, respectively, clients and ADNSs, and the ADNS-facing LDNSs are not recursive. Here, client-facing servers load-balance their queries among ADNS-facing servers based on a hash of the queried hostname; ADNS-facing servers send CNAME responses back to the client-facing server, which forward the subsequent query to a different ADNS-facing server due to different hostname.

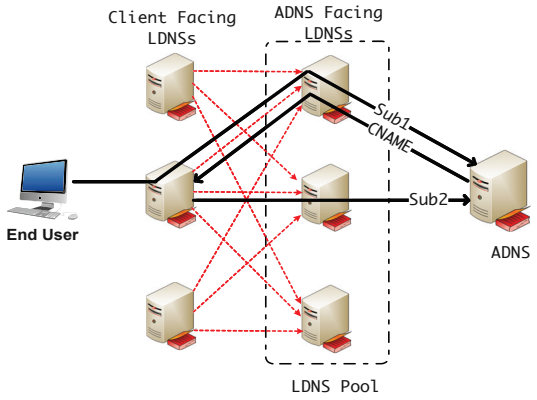


**Figure 18: LDNS Pool**

In this paper we will call such a behavior – for the lack of a better term – the *multiport behavior* and LDNS IP addresses showing together within the same interactions *LDNS pools* to indicate that they belong to the same multiport host or a server farm.

In an attempt to remove fringe scenarios involving rare timeout combinations, we only considered LDNSs $L_1$ and $L_2$ to be part of a pool if (1) the sub1 request for a given client came from $L_1$ while sub2 request for the same client came from $L_2$; and (2) the sub2 request from $L_2$ came within one second of the sub1 request from $L_1$.

Using the above filter, we consider the prevalence of multiport behavior. We found 5,105,467 cases of such behavior representing 407,303 unique LDNS multiport IP address pairs and involving 36,485 unique LDNS IP addresses, or 13% of all LDNSs in our trace. Furthermore, 1,924,359 clients (17% of all clients) were found to be *directly involved* in LDNS multi-port behavior (i.e., observed to have sub1 and sub2 requests within the same interaction coming from different LDNS IP addresses), and over 10M clients – 90% of all the clients in our trace – were *associated* at some point with an LDNS belonging to a pool. Overall, the 13% of LDNSs with multiport behavior were the busiest – they were responsible for over 90% of both sub* queries and subsequent HTTP requests. We conclude that multiport behavior is rather common in today's Internet.

Such significant occurrence of multiport behavior warrants a closer look at this phenomenon as it may have important implications for DNS-based request routing. Indeed, if LDNS pools always pick a random LDNS server to forward a given query, the entire pool and all clients associated with any of its member LDNSs should be treated as a single LDNS cluster. If, however, the LDNS pools attempt to preserve client affinity when selecting LDNS servers (i.e., if the same client tends to be assigned the same LDNS for the same hostname resolution, as would be the case with hash-based assignment sketched earlier) then individual LDNSs in the pool and clients associated with these individual LDNSs could be treated as separate LDNS clusters. A careful investigation of LDNS pools is an open issue for future work.

# 9. DISCUSSION: IMPLICATIONS FOR WEB CONTENT DELIVERY

This section summarizes the implications of our findings for Web platforms that employ DNS-based demand distribution, such as CDNs. Obviously, these lessons were derived from the study of one busy consumer-oriented Web site. While we believe this Web site is typical of similar informational sites, sites of different nature may need to re-evaluate these lessons, in which case our study can serve as a blueprint for such an assessment. The implications discussed here are necessarily qualitative; they follow logically from our findings but each would have to be carefully evaluated in a separate study in the specific target environment.

First, despite a long-held concern about the hidden load problem of DNS-based demand distribution, this is not a serious issue in practice for all but a small fraction of local DNS servers. For most LDNSs, the amount of hidden load – while different from one LDNS to the next – appears small enough to provide sufficiently fine granularity for load distribution. Thus, a proper request routing could achieve a desired load distribution without elaborate specialized mechanisms for dealing with hidden load such as [5, 6].

Second, due to their relatively small number, the exceptions to the above finding ("elephant" LDNS clusters) can be identified, tracked and treated separately, perhaps even by semi-automated policy configuration. This is especially true for the very largest elephants as they appear to be geographically compact: even though these clusters contain tens of thousands clients, their clients are mostly situated within a hundred miles from their LDNS. Thus, these clusters both benefit significantly from being served from a proximally optimal location in the platform *and* are not amenable to being shifted between locations using DNS resolution, due to their large hidden load. More fine-grained demand distribution techniques, such as L4-7 load balancers or HTTP or RTP redirection might be needed.

Third, there is a large variation in the compactness of LDNS clusters, both in terms of geographical distribution of their clients and autonomous system sharing between the clients and the LDNS in the cluster. This provides rich opportunities for improved request routing policies. For instance, the ADNS of the platform can try to "pin" compact LDNS clusters to be served from the respective optimal locations in the platform, while resolving any load imbalances within the global platform by re-routing requests from non-compact clusters to the extent possible. The specific policies must be worked out; however, the amount of diversity in terms of cluster compactness at a large range of cluster sizes makes this a promising avenue for improving efficiency of a Web platform.

Finally, there has been a shift in client-side DNS setup. The traditional model of a stub resolver at a client host talking to a local recursive DNS server, which interacts with the rest of the DS infrastructure, no longer applies to vast numbers of clients. Many clients appear to be behind middleboxes, which masquerade as both a client and its LDNS to the rest of the Internet. Also common are complex setups involving layers of resolvers with shared state, which we called "LDNS pools". While we find no evidence that the former setup requires special treatment from a Web platform, the implications of the wide deployment of LDNS pools is another direction for further investigation.

# 10. RELATED WORK

This paper explores client-side DNS infrastructure. Among the previous client-side DNS studies, Liston at al. [13] and Ager et al. [1] measured LDNS by resolving a large number of hostnames from a limited set of client vantage points (60 in one case and 75 in the other), Pang et al. [19] used access logs from Akamai as well as active probes, and [7] based their studies on large-scale scanning for open resolvers. Our goal was a broad characterization of clients' LDNS clusters from the perspective of a busy Web site.

Both Ager et al. [1] and Huang et al. [11] compared the performance implications of using public DNS resolvers, such as Google DNS, with ISP-deployed resolvers and found the former to be at significantly greater distances from clients. Further, Huang et al. considered the geographical distance distribution between clients and their LDNSs (Fig. 5 in [11]). Our study found these distances to be significantly greater: while they observed 80% of clients to be within 428km of their LDNS resolvers, over 50% of our clients were over 500 miles (806km) away from their resolvers (cf. Fig. 9). Network proximity of clients to their LDNS was considered in [23] and [15]. Our measurement technique is an extension of [15], which we augmented to allow measurement of LDNS pools.

Bermudez et al. proposed a tool that combines a packet sniffer and analyzer to associate content flows with DNS queries [3]. This tool is targeted to operators of client-side access networks, in particular to help them understand which content comes from third-party platforms, while our approach is website-centric, with the goal of characterizing LDNS clusters to inform DNS-based request routing.

As an alternative to the Faster Internet initiative mentioned earlier [8], Huang et al. [10] recently proposed a different method to inform Web sites about the clients behind the LDNS. This proposal does not require changes to DNS and instead modifies client applications, which are presumably more amenable to changes.

# 11. CONCLUSION

This paper investigates clusters of hosts sharing the same local DNS server ("LDNS clusters"). Our study is done from the vantage point of a busy consumer-oriented web site and is based on a large-scale measurement over 28 day period, during which our web page was accessed around 56 million times by 11 million client IPs.

We found that among the two fundamental issues in DNS-based network control - hidden load and client-LDNS distance, hidden load plays appreciable role only for a small number of "elephant" LDNS servers while the client-LDNS distance is significant in many cases. Further, LDNS clusters vary widely in both characteristics, and the largest clusters are actually more compact than others. Thus, a request routing system such as a content delivery network can attempt to balance load by reassigning non-compact LDNSs first as their clients benefit less from proximity-sensitive routing anyway. We also report on several other important aspects of LDNS setups and in particular observed a wide use of what we called "LDNS pools" that – unbeknown to end-hosts – appear to load-balance DNS resolution tasks.

**Acknowledgement.** This work was supported in part by NSF under grant CNS-0831821.

## 12. REFERENCES


[1] Bernhard Ager, Wolfgang Mühlbauer, Georgios Smaragdakis, and Steve Uhlig. Comparing DNS resolvers in the wild. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 15–21, 2010.

[2] H. Alzoubi, M. Rabinovich, and O. Spatscheck. The anatomy of LDNS clusters: Findings and implications for DNS-based network control. http://engr.case.edu/rabinovich_michael/LDNS_full.pdf.

[3] I. Bermudez, M. Mellia, M.M. Munafò, R. Keralapura, and A. Nucci. Dns to the rescue: Discerning content and services in a tangled web. In *Proceedings of the 12th ACM SIGCOMM Conference on Internet Measurement*, 2012.

[4] M. Casado and M.J. Freedman. Peering through the shroud: The effect of edge opacity on IP-based client identification. In *Proceedings of the 4th USENIX conference on Networked systems design & implementation*, pages 13–13. USENIX Association, 2007.

[5] Michele Colajanni and Philip S. Yu. A performance study of robust load sharing strategies for distributed heterogeneous web server systems. *IEEE Trans. Knowl. Data Eng.*, 14(2):398–414, 2002.

[6] Michele Colajanni, Philip S. Yu, and Valeria Cardellini. Dynamic load balancing in geographically distributed heterogeneous web servers. In *ICDCS*, pages 295–302, 1998.

[7] D. Dagon, N. Provos, C.P. Lee, and W. Lee. Corrupted DNS resolution paths: The rise of a malicious resolution authority. In *Proceedings of Network and Distributed Security Symposium (NDSS)*, 2008.

[8] The global internet speedup. http://www.afasterinternet.com/.

[9] Google Public DNS. Performance Benefits. https://developers.google.com/speed/public-dns/docs/performance.

[10] Cheng Huang, Ivan Batanov, and Jin Li. A practical solution to the client-LDNS mismatch problem. *SIGCOMM Comput. Commun. Rev.*, 42(2):35–41, April 2012.

[11] Cheng Huang, D.A. Maltz, Jin Li, and A. Greenberg. Public DNS system and global traffic management. In *INFOCOM, 2011 Proceedings IEEE*, pages 2615 –2623, 2011.

[12] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. Netalyzr: illuminating the edge network. In *Proceedings of the 10th Annual Conference on Internet Measurement*, IMC '10, pages 246–259, 2010.

[13] R. Liston, S. Srinivasan, and E. Zegura. Diversity in DNS performance measures. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 19–31, 2002.

[14] G. Maier, F. Schneider, and A. Feldmann. NAT usage in residential broadband networks. In *12th Passive and Active Measurement Conf.*, pages 32–41, 2011.

[15] Zhuoqing Morley Mao, Charles D. Cranor, Fred Douglis, Michael Rabinovich, Oliver Spatscheck, and Jia Wang. A precise and efficient evaluation of the proximity between web clients and their local DNS servers. In *USENIX Annual Technical Conference, General Track*, pages 229–242, 2002.

[16] Maxmind GeoIP city database. http://www.maxmind.com/app/city.

[17] OpenDNS - A Technical Overview. http://www.opendns.com/technology.

[18] J. Pang, A. Akella, A. Shaikh, B. Krishnamurthy, and S. Seshan. On the responsiveness of DNS-based network control. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 21–26, 2004.

[19] Jeffrey Pang, James Hendricks, Aditya Akella, Roberto De Prisco, Bruce Maggs, and Srinivasan Seshan. Availability, usage, and deployment characteristics of the domain name system. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet measurement*, IMC '04, pages 1–14, 2004.

[20] I. Poese, B. Frank, B. Ager, G. Smaragdakis, and A. Feldmann. Improving content delivery using provider-aided distance information. In *The 10th ACM Internet Measurement Conf.*, pages 22–34, 2010.

[21] M. Rabinovich and O. Spatscheck. *Web caching and replication*. Addison-Wesley, 2001.

[22] K. Schomp, T. Callahan, M. Rabinovich, and M. Allman. Assessing the security of client-side DNS infrastructure. Submitted for publication, 2012.

[23] Anees Shaikh, Renu Tewari, and Mukesh Agrawal. On the effectiveness of DNS-based server selection. In *INFOCOM*, pages 1801–1810, 2001.