On the Responsiveness of DNS-based Network Control

Jeffrey Pang*, Aditya Akella*, Anees Shaikh†, Balachander Krishnamurthy‡, Srinivasan Seshan*

*Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213-3891
{jeffpang,aditya,srini+}@cs.cmu.edu

†Network Software and Services
IBM T.J. Watson Research Center
Hawthorne, NY 10532-2134
aashaikh@watson.ibm.com

‡AT&T Labs – Research
Florham Park, NJ 07932-0971
bala@research.att.com

ABSTRACT

For the last few years, large Web content providers interested in improving their scalability and availability have increasingly turned to three techniques: mirroring, content distribution, and ISP multihoming. The Domain Name System (DNS) has gained a prominent role in the way each of these techniques directs client requests to achieve the goals of scalability and availability. The DNS is thought to offer the transparent and agile control necessary to react quickly to ISP link failures or phenomenon such as flash crowds.

In this paper, we investigate this assumption with the objective of quantifying the degree of responsiveness that can be expected from DNS. We use a combination of Web and DNS access measurements from several busy Web sites, as well as a large content distribution network, to characterize the behavior of end-systems and local DNS servers in terms of their adherence to DNS-based controls. Our results suggest that DNS is at best a coarse-grained mechanism, and poorly suited for applications, such as route control, which require quick response to link failures or performance degradations. We then propose several proactive techniques that, when deployed in cooperation between large content providers and important clients, have the potential to improve the responsiveness of DNS-based control.

Categories and Subject Descriptors: C.2.2 [Computer Systems Organization]: Computer-Communication Networks–*Network Protocols*

General Terms: Measurement

Keywords: DNS, time-to-live, network control

1. INTRODUCTION

With the growing popularity of Internet services and applications, large content and service providers have turned to a number of distribution techniques to improve their scalability, availability, and performance. For example, in the face of an ever-increasing request rate, some popular Web sites deliver their content via redundant mirror sites, or by outsourcing delivery to alternate servers in a content distribution network (CDN). Many large data centers and enterprises also rely on multihomed connectivity, coupled with intelligent route control, to improve resilience to network or link failures while optimizing performance and bandwidth costs.

IMC'04, October 25–27, 2004, Taormina, Sicily, Italy.

A common requirement in each of these approaches is the need for tighter *network control* over client access to the application. For example, in a CDN or mirrored site deployment, clients must be directed to servers that are available and offer good response time. Similarly, when employing route control with multiple ISP connections, directing packets over the correct provider link is crucial to extract the performance and reliability benefits of multihoming.

Over the last few years, the Domain Name System (DNS) [1] has emerged as a common approach for this type of network control. The appeal of DNS-based techniques arises from its ubiquity and transparency. By leveraging the DNS, content providers can provide a "late binding" that controls which IP address is returned to the client during the name resolution operation. This returned IP address, in turn, determines which server is contacted, or which ISP is used. Inherent in this approach is the assumption that DNS-based control responds quickly enough to unexpected conditions, such as link failures, flash crowds [5], or increased congestion.

Responses to name resolution requests have an associated time-to-live (TTL) value that determines how long the response should be cached by the client's local nameserver. Setting the TTL to a very small value (e.g., 10 seconds or even zero) forces clients to resolve the IP address frequently, thus providing fast response. In practice, however, this is complicated by the behavior of the wide variety of applications and DNS servers deployed in the Internet. Many applications perform their own internal DNS caching that does not adhere to the expected behavior, and some implementations of local DNS server (LDNS) software have been reported to not adhere to the specified DNS TTLs [11]. This would imply that clients behind those LDNSes will continue to connect via the failed link, or access the more loaded mirror site.

In this paper, we consider the degree of responsiveness that can be expected from DNS-based network control in practice. We base our analysis on empirical observations of client access behavior for several large Web sites hosting major sports events, as well as requests from a wide variety of LDNSes contacting authoritative DNS servers in a large CDN. Our results show that, while a majority of clients and LDNSes honor DNS TTLs, a significant fraction does not (up to 47% of clients and LDNSes collectively, and 14% of LDNSes in our measurements). Moreover, those that violate TTLs do so by a large amount, in excess of 2 hours.

Our findings suggest a need for mechanisms to improve DNS responsiveness, since we expect Internet services and applications to continue to rely on DNS for network control, in addition to its traditional distributed database function. Hence, we propose several architectural techniques to increase DNS agility that are most useful for deployment by performance- or availability-sensitive application providers in cooperation with specific users.

Previous work has addressed other issues with DNS-based control, focusing on the context of server selection in CDNs. Some examples include: the impact of small TTLs on response time [12]; the effect of DNS lookups on overall client-perceived response time [9]; the question of accuracy due to the proximity of clients and their LDNSes [10, 12]; and the effects on DNS cache hit rates [6] from small TTLs. Our contribution is the measurement and quantification of the level of *responsiveness* offered by the DNS. We view this as an important issue, given the continued growth in the number of services and vendor products using DNS as a control mechanism.

In the next section we provide a brief overview of some of the related research on DNS. Section 3 follows with a description of our measurement data collection methodology. In Section 4 we present our observations and analysis, and follow with proposals for improving the responsiveness of DNS-based control in Section 5. Section 6 summarizes the paper.

2. RELATED WORK

There has been considerable work in the DNS arena ranging from exhaustive characterization on a local and wide-area level to examinations of (mis)use of DNS for specific applications. Traffic to the root DNS servers, especially unnecessary traffic [14], has also been characterized. The CAIDA team has examined the impact of DNS caching software [15] on load at upper levels of DNS hierarchy. In [6], the authors collect local area traces of DNS and application traffic to characterize DNS performance and caching behavior. Proposed modifications to DNS to improve Web performance include piggybacking HTTP responses in DNS replies [7], and having LDNSes renew cache entries proactively [2].

The notion of leases arose in the context of Web cache coherency to reduce staleness in caches. Recent work on adaptive leases [3] discusses ways by which caches can avoid having to constantly poll to reduce risk of staleness primarily by promising to flush cached objects upon lease expiry or when proactively notified by the origin server. We draw upon this idea to examine ways by which Web content owner or a CDN can notify cooperating LDNS to flush their caches.

3. DATA COLLECTION METHODOLOGY

In order to measure the responsiveness of DNS-based control, we should ideally observe how clients respond to DNS changes that update the name-to-address mapping (i.e., `A` records) for a particular domain). This requires, for example, access to application logs at various servers to track the access patterns of clients, along with information about the timing of DNS updates. This would allow us to measure how quickly clients respond to DNS changes. The challenge is to identify data sources from which both DNS *update* logs and application logs are available, and also where DNS entries are updated often enough to collect a reasonable set of observations of how clients react to the DNS.

Since it is difficult to obtain such coordinated data, we use observations of specific behavior from several different data sets to collectively infer DNS responsiveness. In particular, we focus on client and LDNS behavior in terms of adherence to DNS TTLs. Our approach is to identify the frequency with which clients and LDNSes use cached DNS records beyond their specified TTL to access an application or another DNS server. We refer to this behavior as *TTL violations*. If violations are widespread, we can infer that DNS techniques for network control are not well-suited for situations that require fast response. On the other hand, if the number of violations is small, it suggests that DNS-based control can indeed provide good responsiveness.

Below, we describe the characteristics of our measurement data sets, and how we use them in the analysis.

3.1 Observations from large Web events

Our first data set consists of cache logs from a distributed hosting infrastructure that serves content for a number of Web sites. We collected logs from three large sporting events with worldwide audiences that were held in April, June, and July, 2003 and hosted on this infrastructure. During each event, when the request rate was very high, the authoritative name servers directed all clients to the set of distributed caches with a 10 minute TTL. At the conclusion of each event, they were "archived" by updating the name servers to direct clients to lower capacity origin servers. We combine the cache access logs with an administrator log containing timestamps indicating when updates to the DNS are made to archive each event.

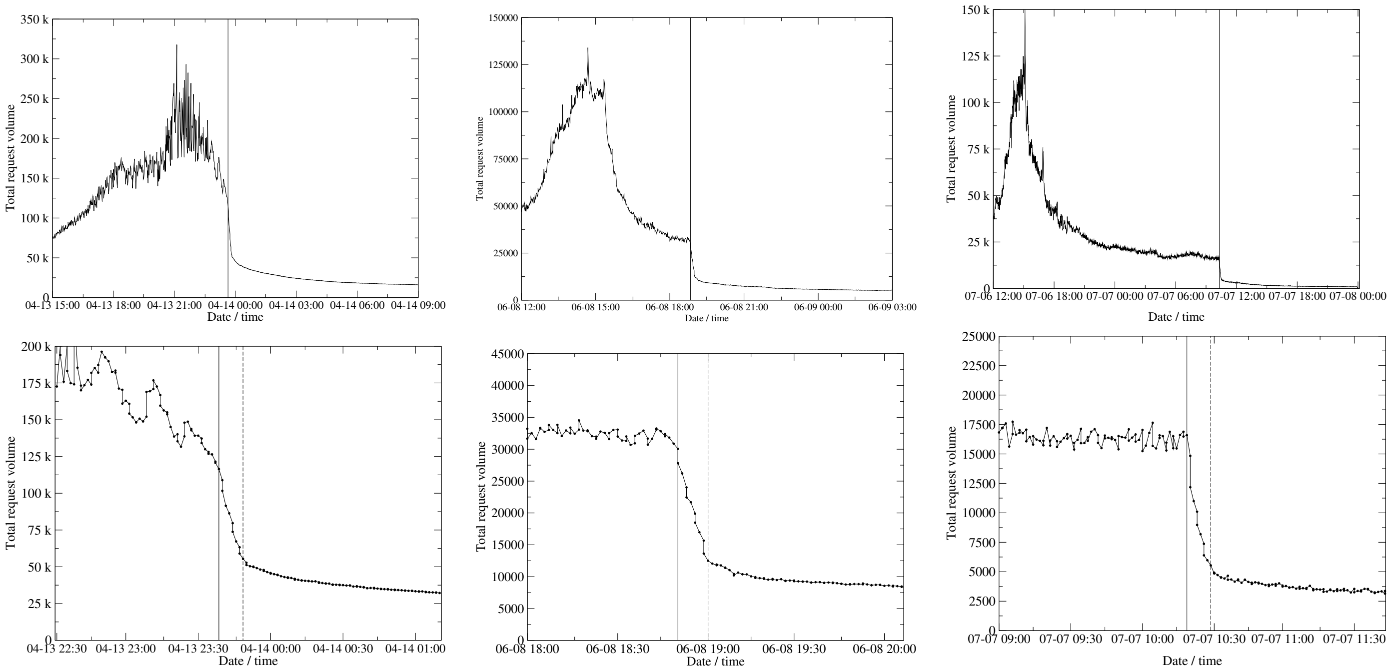
After combining the access logs from all cache sites, we extracted a roughly 2-day long segment for each event near the time at which the corresponding DNS update took place. Ideally, all traffic to the caches should subside 10 minutes (i.e., the TTL period) after the DNS update to archive the event. In our analysis we focus on client requests that continue to arrive at the cache sites after the DNS is updated and the TTL expires. We count these requests as TTL violations, and further classify them according to how long beyond the TTL expiry they arrive at the caches, and the network location of the clients that originate such requests. We also allow a 30 second "grace time" after the DNS update log timestamp to account for any delay in updating the DNS, though we believe this is quite conservative. As a result, we may underestimate the number of violations. Note that since we can only view client requests, we cannot distinguish between violations caused by client applications and those caused by noncompliant LDNS servers. Also, our data for each event extends approximately a day after the TTL expiration, thus we cannot count violations beyond one day.

3.2 LDNS behavior in a large CDN

The second data set consists of measurements from servers belonging to the Akamai CDN. The CDN employs a two-level nameserver hierarchy to achieve fine-grained client redirection and load balancing. By analyzing this data set, we aim to characterize the effectiveness of DNS referrals, and adherence to TTLs on NS records, both of which are crucial parts of a DNS-based network control infrastructure.

In this data set, we first collect DNS request logs (containing `A` requests) at the "high-level" name servers of the CDN for the duration of a day. These logs contain the IP address of the requesting LDNS server, along with a summary of the request made (e.g., the requested domain name, timestamp, etc.). In response to these requests from the client LDNS , the high-level name servers return a referral, typically a list of NS records for "low-level" CDN name servers, which are deemed to be close to the requesting LDNS servers. The LDNS server then sends its request to one of these low-level servers (typically the first on the list). These NS records could each have different TTLs, typically, they are 30, 45, or 60 minutes.

We pick a random sample of 100,000 LDNS IP addresses appearing in the high-level request logs, out of a total of approximately 1.2 million (about 8%). For each LDNS IP from the 100,000 chosen, we track the DNS requests it makes, if any, at the low-level name servers returned to it by the high-level servers. We only tracked 64,611 of the 100,000 since the others did not appear in the low level request logs within the time window we tracked. In the ideal case, after the TTL on the low-level record expires, the LDNS must make a fresh request to the high-level server. Therefore, we check



(a) April 2003

(b) June 2003

(c) July 2003

Figure 1: Traffic volume near a DNS change for three large Web-hosted events. The bottom row of graphs zoom in on the time around the DNS update and TTL expiry. Timestamps on the x-axis are EDT.

to see if the LDNS continues to approach (i.e., send requests to) a given low-level name server even after the NS record it holds for the name server has expired, indicating a violation of the TTL.

Our logs were collected on March 3, 2004 between 1AM and 10PM. We limit our analysis to resolution requests for names that represent static image content, e.g., domain names of the form `aXX.img.CDN.net`. Incorporating requests for other CDN names into our analysis may yield different values for the number of violating LDNSes.

4. ANALYSIS RESULTS

In this section, we describe observations made from the Web event and CDN access measurements and analysis. From these results, we can observe the extent to which clients and LDNSes in our data sets adhere and respond to DNS TTLs, and draw some inferences about the responsiveness of DNS-based control.

4.1 Observations from Web Events

In Figure 1, we show the aggregate request volume to all caches during the time near the end of the three events. These requests were grouped into 1-minute intervals. The top graphs show a clear peak occurring at the end of the event, followed by a period of relatively constant and sustained traffic until the DNS update (marked by the solid line). In each case, the effect of the DNS change is dramatic, causing a sharp drop in requests coming to the cache locations as clients are redirected to the archive servers.

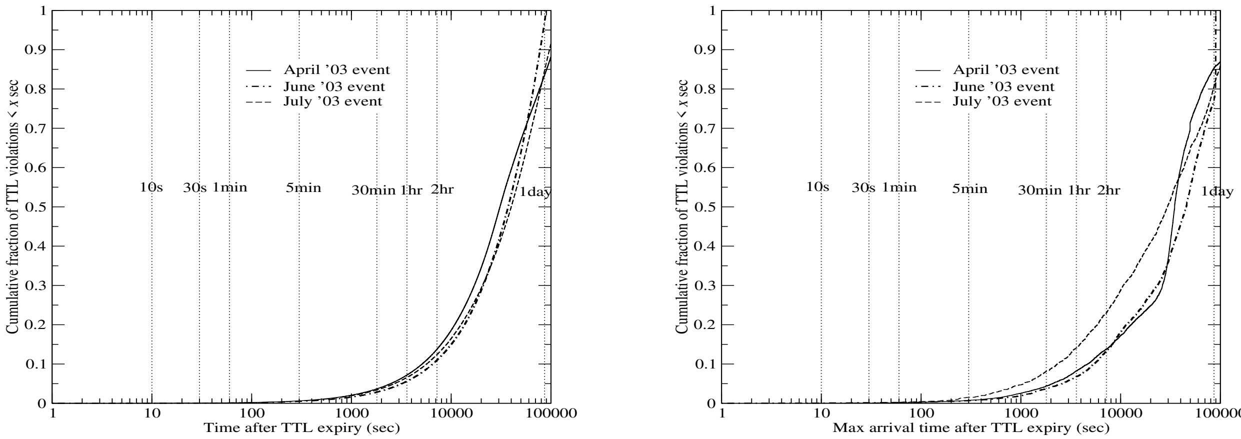
The bottom graphs in Figure 1 zoom in on the portion of the trace near the DNS update. In these graphs, the solid line denotes the time of the update and the dashed line is the time when the 10 minute TTL expires. Requests arriving after the TTL expiry are considered to be in violation in our analysis (subject to the 30 second grace period). Between the update time and the TTL expiration, the request volume decreases rapidly by roughly 53%, 60%, and 67% for the April, June, and July events respectively. However, the remaining one half to one third of the traffic decays very slowly over a long period, which we discuss in more detail below. We can see that while more than half of the client requests can be shifted away quickly, the remaining requests using the stale DNS entries is significant.

In Figure 2(a), we plot the distributions of the extent of DNS TTL violations over all requests arriving after the TTL expiration for each event. A few specific times are annotated on the graph, such as 10 seconds, 30 seconds, up to 1 day after the TTL expiration. The graph shows that the bulk of the violations are very long, between two hours and a day. In fact, the requests may arrive even long after a day, but our access logs extend only roughly a day past the DNS update time. Since the duration of the majority of violations is so long, DNS would provide very little control over clients that do not honor the TTLs. That is, there are very few cases in which clients violate the TTL by only a small amount. This effect is consistent across all three events.

Figure 2(b) further illustrates this effect. It plots a distribution of the maximum observed TTL violation for each client that makes a request after the DNS update. So, if a particular client makes several requests after the TTL expires (i.e., in violation), this graph captures the last time a request from the client was observed. Again, since our traces do not extend beyond one day after the DNS update, the distribution is truncated to roughly a day. Similar to Figure 2(a), we see that most clients that violate the TTL continue to use DNS entries well after they expire. The graph shows that 75–85% of the clients violate the TTL by more than 2 hours.

4.2 LDNS accesses in a large CDN

In Figure 3(a), we show a CDF of the length of the time for which local DNS servers cache NS records with stale TTLs (i.e., the duration of a TTL violation). Notice that about 86% of the random sample of LDNSes observed in the high-level name server trace do not exhibit any violation of the NS record TTL. About



(a) All violations (b) Maximum violation

Figure 2: CDF of the request arrival time in violation of the DNS TTL (a) and the maximum arrival time (b), for each event. Time on the x-axis is on a logarithmic scale.

2% of the LDNSes made exactly one request to the top-level name server and did not return during the course of our log collection. As a result, we could not determine the duration of their TTL violation (as such, the CDF does not reach "1" on the y-axis).

In Figure 3(b), we show a similar CDF for just the violating LDNSes (which constitute about 14% of all LDNS servers). Notice that nearly 70% of the noncompliant LDNSes show a violation of TTLs in excess of 1 hour. Nearly 25% of exhibit violations of more than 5 hours in length. These observations are roughly similar to those made for the Web Event dataset in Figure 2(b).

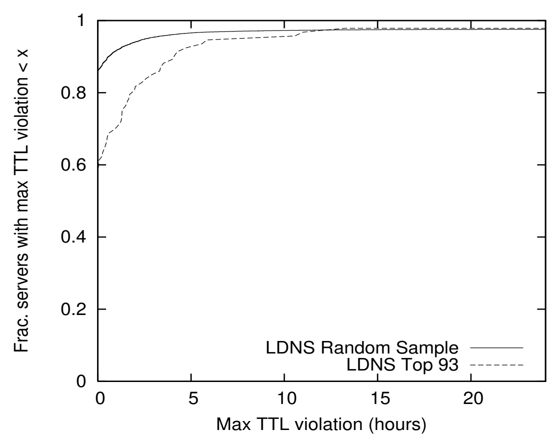
However, the above characterization is biased in favor of LDNSes making very few total requests. That is, we are likely to report some of these LDNSes as adhering to TTLs while, in reality, they may not. This is because our analysis depends heavily on the number and frequency of the requests made by an LDNS in order to classify it as a violator or not. For example, an LDNS server that consistently violates TTLs by about an hour, but has a request rate of once every 2 hours in our low-level trace, is likely to be identified as an TTL-adhering server.

To address this issue, we also plot the CDF of the TTL violation duration for the top 93 LDNSes ranked by the volume of requests they generate to the high-level name server logs in both Figures 3(a) and (b)[1]. Notice in Figure 3(a) that more than 37% of these violate TTLs. Of these, about 15% exhibit TTL violations in excess of 5 hours.

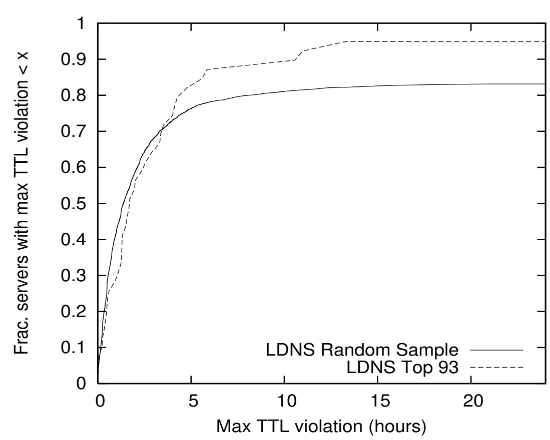
Finally, we characterized the LDNSes according to their network-aware clustering (NAC), which groups IP addresses that are close together topologically and likely under common administrative control [8]. We are interested in whether noncompliant LDNSes are more likely to appear in small networks in which addressing DNS configuration or deployment issues may be easier than in larger ones. Hence, our analysis considers whether clusters of particular sizes have relatively higher fractions of non-compliant LDNSes.

We first collect a large list of all LDNSes observed in both the high-level and the low-level DNS logs, and group them according to their NACs using BGP tables collected by the Route Views project at roughly the same time as our LDNS logs [13]. We do the same for the violating LDNSes. Finally, we classify the LDNSes (both violators and non-violators) according to ranges of the prefix length of their NAC (e.g., the prefix length of the NAC `9.0.0.0/8` is 8).

[1] Although we collected the top 100 requesting LDNSes, 7 of these did not appear in our low-level logs, primarily because we did not have access to all low-level server logs.



(a) All LDNSes



(b) Violating LDNSes

Figure 3: CDFs of the duration of the TTL violation on NS records by LDNSes. The solid line shows the CDF for the random sample of LDNSes; the dashed line shows the CDF for the top 93 requesting LDNSes.

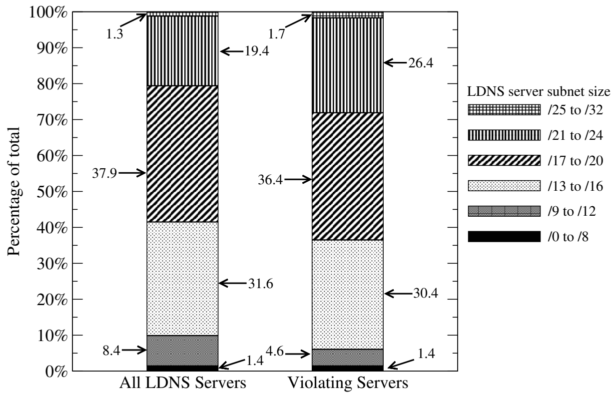


Figure 4: Relative distribution of all LDNSes (left bar) and of violating LDNSes (right bar) according to the size of their corresponding NAC.

The results for this characterization are shown in Figure 4. We see that LDNSes in NACs of lengths 21–24 constitute about 19% of all LDNSes, but 26% of all violating servers, suggesting that these small clusters have a relatively higher fraction of violators. On the other hand, LDNSes in NACS of lengths 9–12, constitute about 8% of the servers, but only 4% of the violators. NACs of lengths 13–16, and 17–20, have a more even distribution of violators.

Our measurements, though based on a limited sample, suggest that the relative proportion of non-compliant LDNSes is higher in smaller NACs (especially in NACs of lengths 21–24). Changing the operation of LDNSes in these smaller networks could eliminate a substantial number of non-responsive local DNS servers.

4.3 Additional findings

A natural question that arises from our analysis is why the number of violations is so significant. One intuition may be that particular noncompliant DNS server implementations are primarily responsible. Or perhaps most of the violations can be traced to a relatively small number of misconfigured networks. In this section, we describe some additional analyses to consider these questions.

DNS implementations. An obvious place to look for root causes of TTL violations is in DNS server implementations. To perform this analysis, we combined our list of violating LDNS servers with a list of LDNS addresses gathered from earlier measurements of DNS implementations taken in October 2003[2]. Our results are limited to Berkeley Internet Name Domain (BIND) implementations, which is widely used in the Internet [4]. We found 11,744 common LDNS addresses for which a BIND version response was available. Of these, 2237 addresses belonged to LDNSes that violated TTLs according to our observations. In Table 1, we list the top 10 BIND implementations over all LDNSes and also the top implementations of violating LDNSes. The table omits the count of DNS error messages returned in response to the version queries (e.g., `NOTIMP`, `REFUSED`, `SERVFAIL`, and `FORMERR`). These responses, particularly `NOTIMP`, often indicate a non-BIND implementation such as Microsoft Windows DNS or TinyDNS.

From Table 1, there does not seem to be a single distribution of BIND that is used by a large majority of violating LDNSes. It is therefore difficult to argue from our data sets that a single software implementation is responsible for TTL violations.

[2]It is possible that the BIND version discovered in October 2003 for a particular address was changed by March 2004.

All LDNSes		Violating LDNSes	
BIND version	count	BIND version	count
9.2.1	2199	9.2.1	248
8.2.3-REL	712	8.2.3-REL	148
9.2.2	378	8.3.4-REL	114
8.3.4-REL	339	9.2.2	64
9.1.3	294	8.3.3-REL-NOESW	55
9.2.0	276	8.2.2-P5	55
8.3.3-REL	219	8.3.3-REL	50
8.2.2-P5	210	8.2.2-P5	42
8.3.3-REL-NOESW	183	8.2.4-REL	30
8.2.4-REL	165	9.1.3	29

Table 1: LDNS BIND versions

Identifying noncompliant networks. We performed a simple enumeration of client networks to gain some initial insight into whether a few clients or networks were primarily responsible for observed TTL violations in our Web access data. For each of the events, we clustered the IP addresses of violating clients into class C networks (e.g., /24 clusters) and ranked them according to how many unique addresses were observed from the cluster. We then focused on identifying the networks of the top few violating clusters to see if any patterns emerged. Note that this approach is limited by the client workload; our identification of the top violators is subject to the popularity of the Web events among clients in each cluster. Nevertheless, a few interesting observations arose which we discuss below.

In two of the three events, the top violating clusters belonged to the Web crawler of a popular search engine. This indicates that crawlers continue to visit live Web pages without re-resolving the corresponding hostnames. We also found a few cases of specific regional networks that were responsible for many violating clients. In the June 2003 event, for example, the top 13 clusters belong to two networks, a provincial network in Asia and a dial-up ISP in Europe. Another notable finding from the April 2003 event was the relatively large number of noncompliant requests generated by dial-up and broadband subscribers of a single U.S. ISP. Finally, we found client addresses belonging to multiple sites of the same enterprise network among the top violating clusters in all three event logs. The last few examples are particularly interesting, since they imply that DNS TTL violations may in fact be due to misconfigurations (or optimizations) in specific networks.

5. ARCHITECTURAL SUPPORT FOR PROACTIVE DNS

Our previous observations suggest that DNS has limited ability to provide fine-grained network control. However, as we argue below, it can still be useful to attain network control in *cooperative settings*. For example, content or application providers can deploy DNS-based control mechanisms in cooperation with customers who stand to benefit. These customers may be willing to modify their DNS infrastructure to enable performance enhancements such as dynamic server selection or route control.

Ideally, the content provider can simply set the TTL on A records aggressively and advise its customers to ensure that their LDNS servers and client applications obey these TTLs. As earlier work [6] has shown, lowering TTLs on `A` records in this manner does not significantly reduce DNS cache hit rates, or cause a large increase in wide-area DNS traffic. Therefore, this is a viable option for customers to consider if they are promised better performance or availability in return.

In addition to simple TTL-based mechanisms, it is possible to achieve additional control in cooperative settings by establishing out-of-band negotiation channels between the content or service providers and the customer LDNS servers. Below we outline two approaches that employ this idea.

Push-based invalidation. In this approach, the ADNS and LDNS servers negotiate out of band during the initial exchange of DNS request and response. Modifying the ADNS server is relatively easy, since it is under the control of the content publisher, who is interested in using DNS-based control. Customers modify their LDNS servers to accept *invalidations* from the ADNS. Further, these customer LDNS servers must act on the invalidations by flushing out their caches. Such an invalidation from the ADNS can trigger a new resolution or can include an alternate address that the LDNS should use until the next communication. Again, the participating set of LDNSes would likely belong to large, "high-end" customers who would gain increased availability and performance from such an approach. The clients in these customer networks are more likely to reach lightly loaded mirror sites of the content publisher and should be able to fetch content faster.

Adaptive leases. The final approach requires tighter integration between the ADNS and LDNS, and also entails more explicit cooperation. In this approach, the ADNS will partition the list of LDNS servers into different classes based on their request frequency and the importance of the client base behind those LDNS servers. Customers in the "high-volume" class (either in terms of traffic or the revenue they generate) should be willing to accept leases and periodically renew them.

During the duration of the lease, the ADNS server will support fine-grained invalidations of resource records. These invalidations could be triggered by the perceived need at the ADNS to force its clients to employ a different address. Alternately, the LDNS can poll the ADNS for updates in the records. The lease renewal period will likely depend on the relative importance of the LDNS but it can be influenced by a variety of other factors, such as the expected frequency of changes in the performance of the path between the publisher and the client. The lease can be communicated to the LDNS via a resource record (RR) with the duration of the lease being set as the TTL for the record, for example.

6. SUMMARY

In this paper, we consider the degree of responsiveness that can be expected from DNS-based network control techniques such as server selection in CDNs or link selection in multihomed end-networks. We collect measurements of client access behavior for large Web sites, as well as requests from LDNSes accessing nameservers in a large CDN. Our results show a majority of clients and LDNSes honor DNS TTLs, but a significant fraction does not. For example up to 47% of Web event clients, and 14% of LDNSes in our measurements do not adhere to DNS TTLs. Moreover, those that violate TTLs do so by a large amount, in excess of 2 hours. We also suggested several architectural techniques, including proactive invalidation and adaptive leases, that can be deployed cooperatively between application providers and their customers to improve the responsiveness of DNS-based control.

As future work, we plan to conduct a more active study of LDNS behavior, for example using probes to trigger DNS lookups to gauge their adherence to TTLs. We also intend to further investigate causes of noncompliance, and develop and evaluate our initial architectural techniques.

Acknowledgment

We are very grateful to Roberto De Prisco (Akamai), Bruce Maggs (Akamai and CMU), and Herbie Pearthree (IBM Global Services) for their assistance in obtaining log data for this study. We also thank Oliver Spatscheck, Michael Rabinovich, Duane Wessels, and the anonymous reviewers for their valuable feedback on the paper.

7. REFERENCES

[1] P. Albitz and C. Liu. *DNS and BIND*. O'Reilly and Associates, 2001.

[2] E. Cohen and H. Kaplan. Proactive caching of DNS records: Addressing a performance bottleneck. In *Proceedings of the Symposium on Applications and the Internet*, January 2001.

[3] V. Duvvuri, P. Shenoy, and R. Tewari. Adaptive leases: A strong consistency mechanism for the World Wide Web. *IEEE Transactions on Knowledge and Data Engineering*, 5(5):1266–1276, September 2003.

[4] Internet Systems Consortium. ISC BIND. `http://www.isc.org/sw/bind`.

[5] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In *International World Wide Web Conference (WWW)*, May 2002.

[6] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS performance and the effectiveness of caching. *IEEE/ACM Transactions on Networking*, 10(5), October 2003.

[7] B. Krishnamurthy, R. Liston, and M. Rabinovich. DEW: DNS-enhanced Web for faster content delivery. In *International World Wide Web Conference (WWW)*, Budapest,Hungary, 2003.

[8] B. Krishnamurthy and J. Wang. On Network-Aware Clustering of Web Clients. In *Proceedings of ACM SIGCOMM*, Stockholm, Sweden, August 2000.

[9] B. Krishnamurthy, C. Wills, and Y. Zhang. On the use and performance of content distribution networks. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop (IMW)*, San Francisco, CA, November 2001.

[10] 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. In *Proceedings of USENIX Annual Technical Conference*, Monterey, CA, June 2002.

[11] North American Network Operators' Group. Nanog mailing list. `http://www.nanog.org/mailinglist.html`, 1999,2000.

[12] A. Shaikh, R. Tewari, and M. Agrawal. On the effectiveness of DNS-based server selection. In *Proceedings of IEEE INFOCOM*, Anchorage, AK, April 2001.

[13] University of Oregon. Route views project. `http://routeviews.org`.

[14] D. Wessels and M. Fomenkov. Wow, that's a lot of packets. In *Proceedings of Passive and Active Measurement Workshop (PAM)*, La Jolla, CA, April 2003.

[15] D. Wessels, M. Fomenkov, N. Brownlee, and kc claffy. Measurements and laboratory simulations of the upper DNS hierarchy. In *Proceedings of Passive and Active Measurement Workshop (PAM)*, Antibes Juan-les-Pins, France, April 2004.