Measurements and Laboratory Simulations of the Upper DNS Hierarchy

Duane Wessels[1], Marina Fomenkov[2], Nevil Brownlee[2], and kc claffy[2]

[1] The Measurement Factory, Inc.
`wessels@measurement-factory.com`
[2] CAIDA, San Diego Supercomputer Center, University of California, San Diego
`{marina, nevil, kc}@caida.org`

Abstract. Given that the global DNS system, especially at the higher root and top-levels, experiences significant query loads, we seek to answer the following questions: (1) How does the choice of DNS caching software for local resolvers affect query load at the higher levels? (2) How do DNS caching implementations spread the query load among a set of higher level DNS servers?
To answer these questions we did case studies of workday DNS traffic at the University of California San Diego (USA), the University of Auckland (New Zealand), and the University of Colorado at Boulder (USA). We also tested various DNS caching implementations in fully controlled laboratory experiments. This paper presents the results of our analysis of real and simulated DNS traffic. We make recommendations to network administrators and software developers aimed at improving the overall DNS system.

1 Background

The Domain Name System (DNS) is a fundamental component of the modern Internet [1], providing a critical link between human users and Internet routing infrastructure by mapping host names to IP addresses. The DNS hierarchical name space is divided into zones and coded in the widespread "dots" structure. For example, *com* is the parent zone for *microsoft.com*, *cnn.com*, and approximately 20 million other zones.

The DNS hierarchy's root zone is served by 13 nameservers, known collectively as the DNS root servers. However, the root server operators use various techniques to distribute the load among more than 13 servers. Perhaps the oldest technique is a load-balancing switch for servers at the same site. The most recent technique is IP anycast. We estimate that, by now, there are actually close to 100 physical DNS root servers, even though there are only 13 root server addresses [2].

Just under the root are servers for the Internet's 260-or-so top-level domains (TLDs), such as *.com*, *.nz*, and *.us*. TLD servers are similar to the roots in their requirements, architecture and traffic levels. In fact, many of them are significantly busier. The TLD zones contain referrals to second-level domains (SLDs), and so on.

Why does the Internet need so many root and TLD servers? Is this simply a fact of life, or have we inadvertently created a monster? One of the reasons that we need so many is to carry the normal request load, including a high proportion of unanswerable requests from poorly configured or buggy DNS resolvers. Another is to provide resilience against increasingly common distributed denial of service attacks targeting the system. As the top of the DNS hierarchy becomes more distributed, it is harder for attackers to affect the service.

In this paper, we focus on the behavior of various DNS caching implementations. The behavior of DNS caches is important because they generate almost all of the queries to authoritative (root, TLD, SLD, etc) servers on the Internet. Authoritative servers do not query each other. Stub resolvers should always send their queries to caching nameservers, rather than talk directly to the authoritative servers.

2 Measurements of Live DNS Traffic

For a case study of workday DNS load in academic networks we used the NeTraMet tool [3] to capture samples of DNS traffic (Table 1) at the University of California San Diego (UCSD), USA, the University of Colorado (UCol), USA and the University of Auckland (UA), New Zealand. We configured NeTraMet to select only DNS packets to/from the root and gTLD[3] nameservers; these form a small set of high-level nameservers with known IP addresses. Each DNS sample provides the following information: time (in ms) when a query was sent or a response was received, which root or gTLD server it was sent to, the source host IP address, and a query ID. The UA and UCol samples (but not UCSD) include query types. The most recent samples collected in December 2003 also captured top level and second level domains for each query.

Samples	UCSD Feb 03	U. of Auckland Sep 03	U. of Auckland Dec 03	U. Colorado Dec 03
Query rates to roots, per min	214	29	10	9
Query rates to gTLDs, per min	525	67	76	70
Number of query sources	147	19	42	1
Sample duration, hrs	48	64.5	157.3	157.2
Median response times (from roots)	5–200	60–340	5–290	27–180
% due to 3 highest users	58	85	74	100
% of A queries	n/a	23	70	81

Table 1. Samples of DNS traffic to root and gTLD servers

The number of source hosts addressing the roots and/or gTLDs in the UCSD and UA data is larger than we expected. However, the distribution of queries among sources is highly skewed. Typically, the bottom half of sources produce $< 1\%$ of the overall query load while a small number (3–5) of top sources is responsible for 80% or more of the total traffic. Network administrators need to exercise proper control of those few *high level sources* that dominate external DNS traffic and to optimize the software those source hosts use. At the same time, end user hosts should be configured to send DNS requests to (internal) local nameservers, rather than to root and TLD servers directly.

Typically, the DNS traffic is dominated by A queries (hostname-to-address lookups) as in our December samples. This observation is in general agreement with other measurements (cf. [4], [5]). The first sample of UA DNS traffic was unusual because two

[3] The gTLDs are: *com*, *net*, *org*, *edu*, *mil*, *int*, and *arpa*.

of the three top users generated only PTR queries (address-to-hostname lookups). With the exception of these two sources, A queries constitute 58% of the remaining traffic.

2.1 Response Time and Server Selection

We found that in all samples the root server response time distributions has a long tail, except for G root, which is bimodal. Table 1 shows the range of median root server response times experienced by the busiest query source in each data set. For all nameservers at the same location response time distributions are very similar since the response time correlates with the geographical distance from the point of measurements to a given root server [6]. Nowadays, due to proliferation of anycast root server instances, the exact geographical location of a given root server has become a moot point.[4] Response time actually correlates with the distance to the closest anycast node. For example, the UA nameservers have a median response time of only 5 ms from the anycast F root node in New Zealand.

As we show in the next section, different resolvers utilize different nameserver selection algorithms. For the live data, we studied the probability of selecting a given root server versus its median response time (Figure 1). For each query source, response times were normalized by the minimum response time to the closest root server. When possible, we verified the software being used by hosts at each site. UCol was running BIND8, while both UA nameservers shown were running BIND9. For the UCol and UA#2 sources the probability is approximately inversely proportional to the response time, as expected. However, the UA#1 sent its queries (predominantly PTR) to the root servers in nearly uniform manner ignoring the latency. The UCSD#1 nameserver was using some Windows software and we were not able to obtain any indicative information about UCSD#2. It appears that both UCSD hosts more or less ignore response times when choosing which root servers to use.

2.2 Query Rates, Duplication and Loss of Connectivity

We analyzed our most detailed samples from December 2003 in order to estimate the contribution of repeated queries to the total traffic. Table 2 compares the behavior of A queries to root and gTLD servers sent by the UCol nameserver and by the busiest UA source. For both of them A queries make up about 80% of the total traffic they generate.[5]

For both sources, the bulk of A queries to gTLD servers receive positive answers and, on average, there are 3 to 4 queries per each unique SLD. However, only 2% of A queries sent to root servers by UCol host were answered positively, with an average of 6.7 queries per each unique TLD. This result is in agreement with findings of [7] which showed that only 2% of queries reaching root servers are legitimate. The average repetition rate corresponds to about one query per TLD per day which seems reasonable. At the same time, queries for invalid domains seem to be repeated unnecessarily.

[4] [2] gives locations of anycast nodes for the root servers. Unfortunately there is no straightforward way to determine which server instance any particular DNS reply packet came from.

[5] Next most frequent types of queries are: for the UA source – PTR, 11%, for the UCol source – SOA, 9%.

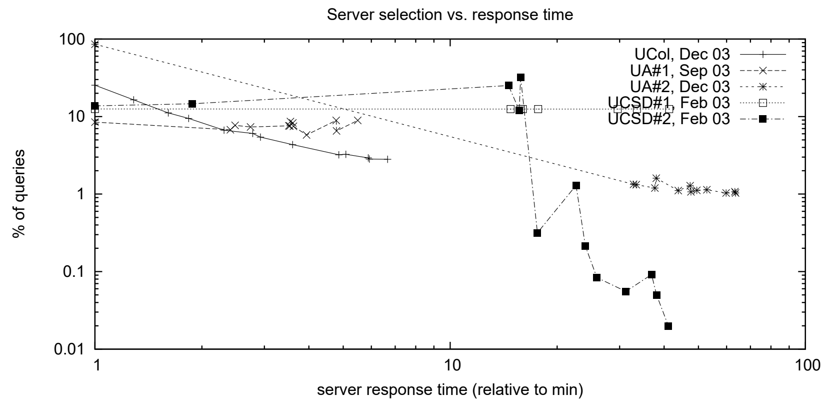

Fig. 1. Distributions of queries among root servers vs. response time. Each point on the plots shows percentage of queries sent to a given root server; the points are plotted in order of increasing response time.

In contrast, the busiest UA nameserver received 77.7% of positive answers from the roots. The average number of queries per TLD is 125.4. For example, there were 5380 queries for *.net* or approximately one query every two minutes. Each one received a positive answer and yet the server continued asking them.

Samples, 6.5 days long	U. Auckland		U. Colorado	
Queries to	roots	gTLDs	roots	gTLDs
positive answers, %	77.7%	94.7%	2.1%	96.3%
av. # of q. per domain	125.4	3.0	6.7	4.4
most frequent	*.net*, 5380	*.iwaynetworks.com*, 2220	*.mil*, 293	*.needsomedealz.com*, 6014
negative answers, %	20.6%	1.9%	97.1%	3.4%
most frequent	*.cgs*, 325	*.coltframe.com*, 134	*.dnv*, 13660	*.jclnt.com*, 7694

Table 2. Statistics of A queries

Samples	U. Auckland		U. Colorado	
Queries to	roots	gTLDs	roots	gTLDs
real traffic	96,759	722,265	86,229	658,784
TTL = 3 h	20,833	306,566	30,243	308,002
TTL = 24 h	10,140	211,914	26,165	229,080

Table 3. Simulated query rates

Our data indicate that repeated queries constitute a considerable fraction of the overall load and that both positive and negative caching can be improved. Using the actual query load observed, we simulated an ideal case when each query would be asked only once: (a) it is not repeated impatiently before the answer comes back, and (b) the answer, whether negative or positive, is cached and the same query is not repeated until its TTL expires. The results (Table 3) indicate that proper timing and caching of queries possibly can reduce the load by a factor of 3 or more. However, this simulation obviously is too optimistic since some repetition of queries to root/TLD servers is unavoidable, for example in a case when queries for *aaa.foo.com* and *bbb.foo.com* immediately follow each other and neither *.com* nor *foo.com* are cached.

Loss of connectivity is another cause of repeated queries. In all our measurements, the loss of packets was very low, typically $< 0.5\%$. However, there was a one hour period of connectivity loss in the University of Auckland data on 6 Dec 03, during which their nameservers sent queries but no answers came back. The query rate increased more than 10-fold, but quickly subsided back to normal levels as soon as connectivity resumed (Figure 2).

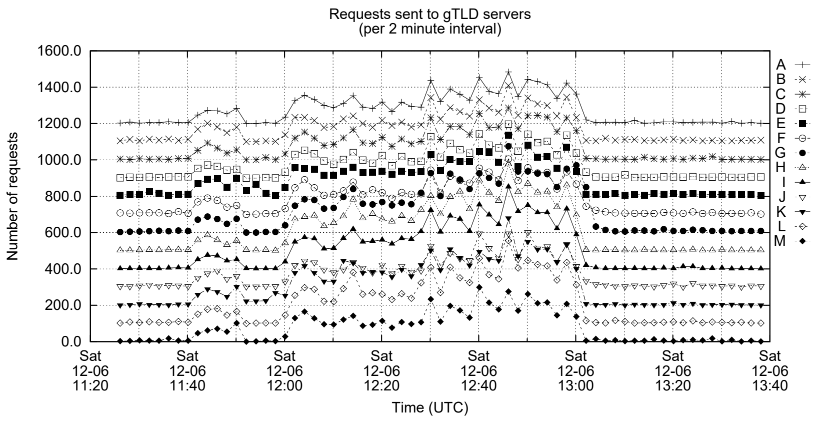

Fig. 2. Query rates to gTLD servers. Each subsequent plot is shifted up by 100. There were two periods of connectivity loss when no answers were received: from 11:40 till 11:50 and from 12:00 till 13:00. During these periods query rates increased considerably.

3 Laboratory Simulations

3.1 Experimental Setup

We set up a mini-Internet in our laboratory to study the behavior of DNS caches. Do they distribute the load as they should? What happens when the cache cannot talk to any root servers? In our setup (Figure 3), three systems on top mimic the authoritative root, TLD, and SLD servers. These systems all run BIND8. Another system, “Cache” in the middle, runs the various DNS caches that we test, one at a time. The bottom system, “User”, generates the test load of end-user queries.

Networking and Addressing. It is somewhat of a stretch to replicate the massive DNS system with only fi ve computers. We use a few tricks to make this a little more realistic. We gave the root, TLD, and SLD authoritative servers multiple IP addresses (13, 254, and 254, correspondingly), bound to the loopback interface. Thus, each server's external interface acts like a router—a way to reach the loopback addresses. The caching nameserver, and the User system have one address each. All systems are attached to a single 100baseTX Ethernet segment. We use FreeBSD's Dummynet feature to introduce simulated packet loss and delays for some tests.

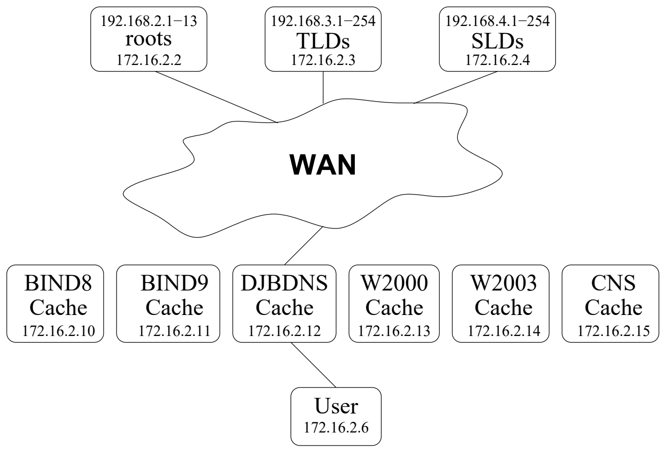

Fig. 3. Network setup for DNS tests. The "User"system sends DNS queries to only one of the caches in each test. The cache recursively resolves queries for the user by talking to the authoritative root, TLD, and SLD servers. For some tests we introduce wide-area packet loss and transmission delays with FreeBSD's dummynet.

Trace Data. To drive the simulation, we created a trace file of DNS requests by collecting hostnames from 12 hours of the IRCache HTTP proxies [8] data. The trace file contains 5,532,641 DNS requests for 107,777 unique hostnames. There are 70,365 unique second-level zones, and 431 top-level zones (many of them bogus). The trace contains many repeated queries as necessary to test the caching part of the DNS cache.

We also take timestamps from the proxy logs and replay the trace at the same rate, preserving as closely as possible the time between consecutive queries. Thus, each test run takes 12 hours to complete.

Zone Files. Our scripts generate BIND zone files based on the contents of the trace file. We do not generate zones for bogus TLDs. Thus, we can also test negative caching.

For the root and TLD zones, we try to mimic reality as closely as possible. For example, we use the same number of nameserver IP addresses and the same TTLs for NS and glue records.[6] To get the real values, we issued the necessary queries out to the Internet while building the zone file data.

We also try to match our simulated SLD parameters to reality. There are too many SLD zones (more than 100,000) and querying all of them would take much too long. Instead, we obtained the necessary values (the number of A records per name and the TTLs for A, NS, and CNAME records) from a random sample of 5000 domains and generated similar random values for the SLD zones in the trace. Unlike reality, each simulated SLD zone has only two NS records. We also determined that about 35% of the names in the trace file actually point to CNAME records, and our simulated zone data mimics this as well.

[6] Glue records are, essentially, A records for nameservers of delegated zones.

Tested Configurations. We tested the following six different DNS caches using their default configuration parameters.

1. BIND 8.4.3
2. BIND 9.2.1
3. dnscache 1.05, aka DJBDNS with CACHESIZE set to 100,000,000
4. Microsoft Windows 2000 v5.0.49664
5. Microsoft Windows 2003 v5.2.3790.0
6. CNS 1.2.0.3

For each cache, we ran tests for six different network configurations:

1. No delays, no loss
2. 100 millisecond delays, no loss
3. linear delays, no loss
4. linear delays, 5% loss
5. linear delays, 25% loss
6. 100% loss

The no-delay/no-loss configuration is simple, but unrealistic. For most real users, the root/TLD/SLD servers are between 30 and 200 milliseconds away (cf. Table 2). The 100ms-delay/no-loss test uses a constant 100-millisecond delay to each root/TLD/SLD server, but with no packet loss. It is also somewhat unrealistic.

In the next three configurations, with so-called linear delays, a nameserver's latency (in ms) is proportional to its order n in the nameserver list: $\tau = 30 + 10n$. For zones such as the root and *com*, which have 13 nameservers, the last one is 160 milliseconds away from the cache. This arrangement provides some pseudo-realistic diversity and allows us to see how often a DNS cache actually selects a nameserver with the best response time. We believe the linear-delay/5%-loss test to be the most realistic among all six configurations.

The final configuration has 100% packet loss. This test mimics a situation when a DNS cache cannot communicate with any authoritative nameservers. Reasons for such non-communication include fire walls, packet filters, unroutable source addresses, and saturated network connections. Live measurements showed that when the cache's queries reach an authoritative nameserver, but the replies do not make it back, the DNS traffic increases (cf. Figure 2).

3.2 Results

General statistics. Figure 4 shows how many queries each DNS cache sent in various tests. For example, the leftmost bar shows that in the no-delay/no-loss test, BIND8 sent a small number of queries to the roots, about 400,000 queries to the TLDs, and about 600,000 to the SLDs. Its total query count is just over 1,000,000.

BIND8 always sends more queries than any of the other caches, primarily because it sends three queries (A, A6, and AAAA) to the roots/TDLs/SLDs for each of the expired nameserver addresses for a given zone. Recall that every SLD zone in our model has two nameservers, while root and TLD zones usually have more. This result implies that

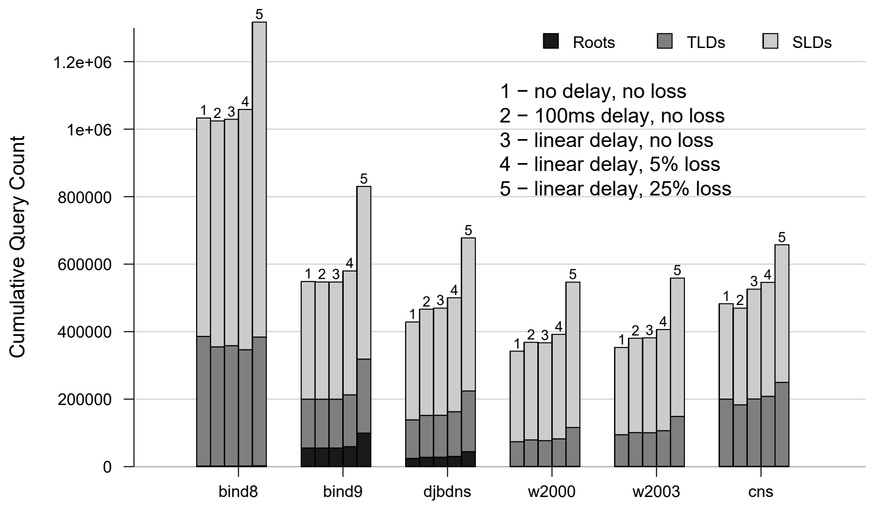

Fig. 4. Cumulative query counts for all caches.

the number of nameservers and their address record TTLs can have a significant impact on DNS traffic.

The BIND9 results show the largest percentage of root server queries. For the no-loss tests, BIND9 sends 55,000 queries to the roots, versus only 1800 for BIND8. The reason for this is because BIND9 always starts at the root when querying for expired nameserver addresses and sends an A and A6 query for each nameserver.

The DJBDNS data also shows a relatively high fraction of root server queries, which is about half of the BIND9 numbers. DJBDNS also starts at the root when refreshing expired nameserver addresses. However, it only sends a single A query for one of the nameservers, not all of them.

The two Windows results are very similar, with slightly more queries from W2003. These two send the fewest overall number of queries for the five test cases shown.

CNS fits in between DJBDNS and BIND9 in terms of total number of queries. Due to its low root server query count, we can conclude that it does not start at the root for expired nameserver addresses. Also note that, like BIND8, CNS sends slightly fewer queries for the 100-millisecond delay test (#2), than it does for the no-delay test (#1).

Root Server Query Counts. Table 4 shows counts of queries to the roots in each test. As discussed above, except for the 100% loss tests, BIND9 and DJBDNS hit the roots much harder than the others. The BIND8 numbers would probably be closer to Windows if it did not send out A6 and AAAA, in addition to A queries.

The 100% loss tests are very interesting. Our trace contains 5,500,000 hostnames and that is how many queries the fake user sends to the cache. Except for BIND9, all other caches become very aggressive when they do not get any root server responses. They send out more queries than they receive. On one hand this is understandable because there are 13 root nameservers to reach. However, it is appalling that an application increases its query rate when it can easily detect a communication failure. The worst offender, W2000, actually amplifies the query stream by more than order of magnitude.

Delays	none	100ms	linear	linear	linear	
Pkt Loss	none	none	none	5%	25%	100%
BIND8	1,826	1,876	1,874	1,899	2,598	37,623,266
BIND9	55,329	55,260	55,256	59,222	99,422	2,564,646
DJBDNS	24,328	27,646	27,985	30,341	44,503	12,155,335
W2000	622	657	663	727	1,020	66,272,276
W2003	693	669	666	709	1,009	39,916,750
CNS	824	831	924	975	1,179	12,456,959

Table 4. Number of Messages sent to roots.

BIND9 is the exception in the 100% loss tests. It actually attenuates the client's query stream, but only by about half. BIND9 is the only DNS caching software that has a nifty feature: it avoids repeat queries for pending answers. For example, if user sends two back-to-back queries (with different query-IDs) for *www.example.com*,[7] most caching resolvers will forward both queries. BIND9, however, forwards only one query. When the authoritative answer comes back, it sends two answers to the user.

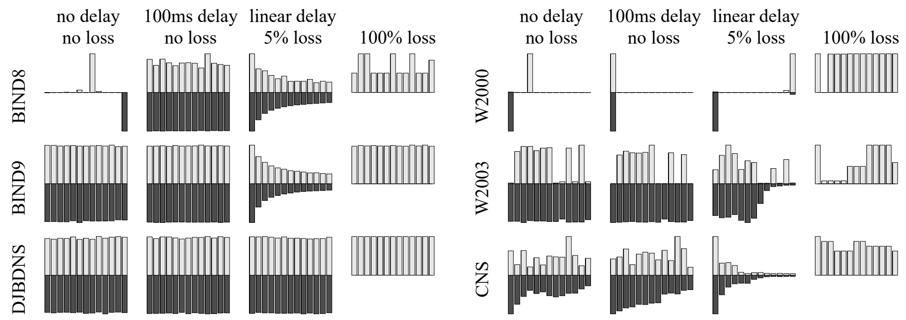

Table 5. Distribution of queries to root and *com* TLD nameservers, showing how the caching nameserver distributed its queries in each test. The upper, lighter bars show the histogram for root nameservers. The lower, darker bars show the histogram for the *com* TLD nameservers.

Root/COM Nameserver Distribution. Table 5 shows how the caches distribute their queries to the root (lighter bars) and to the *com* TLD server (darker bars) for four of the test cases. BIND8 almost uses a single server exclusively in the no-delay/no-loss test. Since that server always answers quickly, there is no need to try any of the others. In the 100-millisecond delay test, however, BIND8 hits all servers almost uniformly. The test with linear delays and 5% loss has an exponential-looking shape. The first nameservers have the lowest delays, and, unsurprisingly, they receive the most queries. The 100% loss test is odd because, for some reason, five of the servers receive twice as many queries as the others.

The BIND9 graphs look very nice. The two no-loss tests, and the 100%-loss test, show a very uniform distribution. Also, the linear-delay/5%-loss test yields another exponential-looking curve, which is even smoother than the one for BIND8.

DJBDNS shows a uniform distribution for all tests. The software does not try to find the best server based on delays or packet loss. This is an intentional design feature.[8]

[7] Of course the answer must not already be cached.

[8] See `http://cr.yp.to/djbdns/notes.html`

Windows 2000 has very poor server selection algorithms, as evidenced by the histograms for the 100ms-delay and linear-delay tests. It selects an apparently random server and continues using it. In the 100%-loss test it did query all roots, except the second one for some reason. The Windows 2003 DNS cache is somewhat better, but also shows strange patterns.

CNS also demonstrated odd server selections. To its credit, the linear-delay/5%-loss case looks reasonable, with most of the queries going to the closest servers. In the 100%-loss test, the selection is almost uniform, but not quite.

4 Conclusion

Our laboratory tests show that caching nameservers use very different approaches to distribute the query load to the upper levels. Both versions of BIND favor servers with lower round-trip times, DJBDNS always uses a uniform distribution, Windows 2000 locks on to a single, random nameserver, and Windows 2003 shows an odd, unbalanced distribution. BIND9 and DJBDNS hit the roots much harder than other caches to avoid certain cache poisoning scenarios. BIND8 and BIND9's optimism in looking for IPv6 addresses results in a significant amount of unanswerable queries.

DNS zone administrators should understand that their choice of TTLs affects the system as a whole, rather than their own nameservers. For example, a BIND9 cache sends two root server queries each time a nameserver address expires. Popular sites can help reduce global query load by using longer TTLs.

Both laboratory tests and live measurements show that caching resolvers become very aggressive when cut off from the DNS hierarchy. We believe that resolvers should implement exponential backoff algorithms when a nameserver stops responding. We feel strongly that more caching implementations should adopt BIND9's feature that avoids forwarding duplicate queries. Other implementations should employ DJBDNS's minimalist strategy of sending only a single A query for expired nameserver glue.

Acknowledgments: Support for this work is provided by WIDE project (`http://www.wide.ad.jp`) and by DARPA grant 66001-01-1-8909.

References

1. Albitz, P., Liu, C.: DNS and BIND. O'Reilly and Associates (1998)
2. RSSAC: Root servers locations (2004) `http://www.root-servers.org/`.
3. Brownlee, N.: NeTraMet - a Network Traffic Flow Measurement Tool (2002) `http://www.caida.org/tools/measurement/netramet/`.
4. Jung, J., Sit, E., Balakrishnan, H., Morris, R.: DNS Performance and the Effectiveness of Caching. In: ACM SIGCOMM Internet Measurement Workshop. (2001)
5. Keys, K.: Clients of dns root servers (private communication) (2002) `http://www.caida.org/~kkeys/dns/2002-08-28/`.
6. T. Lee, B. Huffaker, M. Fomenkov, kc claffy: On the problem of optimization of DNS root servers' placement. In: PAM. (2003)
7. D. Wessels, M. Fomenkov: Wow, That's a Lot of Packets. In: PAM. (2003)
8. IRCache: Information resource caching project (2004) Funded by NSF grants NCR-9616602 and NCR-9521745, `http://www.ircache.net/`.