Root Zone Augmentation and Impact Analysis

Duane Wessels
DNS-OARC
wessels@dns-oarc.net

Geoffrey Sisson
geoff@geoff.co.uk

September 17, 2009

Executive Summary

A number of developments within the last 12 months promise to bring changes to the upper layers of the Domain Name System. In combination, these changes have the potential to radically transform the DNS root zone. DNS-OARC has, under a contract with ICANN, studied the impact of these proposed or imminent changes to the root zone.

One of these changes is the increasing deployment of IPv6 in both the root zone and the TLDs. ICANN has been offering AAAA record publication in the root zone since 2004; however, uptake has been somewhat slow. Five years later (July 2009), 169 TLDs have AAAA records while the other 111 do not.

Another significant change is the advancing deployment of DNSSEC. At present, 10 full-production TLDs are signing their zones. The root zone remains unsigned, though, and DNSSEC-related records have yet to be added to it. However, parties responsible for the management of the root zone say they expect it to be signed by the end of this year.

The final – and perhaps most significant – change addressed by this study is the possibility of introducing new gTLDs. ICANN has proposed a new gTLD program under which the root zone could grow by orders of magnitude.[1]

In this study, we undertake a number of simulations and measurements with BIND and NSD server software and varying zone sizes to better understand how these changes may affect the performance of, and resource requirements for, the root DNS server infrastructure. Our analysis looks at five key areas that would have an impact on operations: zone size, name server reload and restart times, DNS response latency, inter-nameserver bandwidth utilization, and potential increases in TCP usage.

Our analysis of **zone size** focuses on memory usage. As expected, we find that memory requirements increase linearly with zone size. We also find that, for a given number of TLDs, signing the zone increases the memory requirement by a factor of 1.5—2. Additionally, we find that 32 GB of

memory is insufficient for serving a very large root zone (e.g., a signed zone with 10 million TLDs), particularly when using NSD.

The **response latency** measurements find negligible increases (typically less than one millisecond) with NSD. For BIND (9.6.0-P1), however, we find some response time degradation with a large signed root zone (e.g., greater than 100,000 TLDs). With a 100,000 TLD signed zone, BIND drops nearly 30% of all queries sent at a rate of 5000 queries per second. With a one million TLD signed zone, BIND drops over 80%. NSD also begins to show some signs of stress with a very large (4.5 million TLD) zone where over 40% of queries are dropped.

The **reload and restart times** measurements are relatively straightforward and contain no real surprises. Loading and reloading times are generally proportional to zone size. Loading a 1 million TLD signed zone takes 190 seconds with BIND and 227 seconds with NSD.

To measure **inter-nameserver bandwidth** we performed a number of zone transfers between master and slave nameservers. We tested both standard (AXFR) and incremental (IXFR) zone transfer mechanisms. One interesting result of the AXFR test is that an NSD master utilizes 20—30% less bandwidth than a BIND master to send a given zone. To assess the duration of a zone transfer under wide-area network conditions, we introduced simulated packet loss and delays. A zone transfer experiencing 1% packet loss takes more than 2.5 times longer than with no packet loss for any given tested latency.

To explore **increased TCP** at root servers, we replayed real query streams to servers with signed zones. We found that between 0.3% and 0.8% of responses to UDP queries would be truncated, likely causing most these clients to fall back to TCP. This means that root servers can expect to see at least an order of magnitude increase (e.g., from 5 to 50 per second) in queries over TCP when the root zone is signed. Additionally, we found that a large (e.g., one million TLD) signed root zone will likely result in a slightly higher proportion of TCP queries than a signed version of the current one. Finally, we examined data for the `.org` TLD from before and after DNSSEC was deployed and found evidence suggesting that the actual increase in TCP-based queries could be significantly higher than can be forecast by evaluating current DNS traffic patterns.

Contents

Chapter 1

System Setup

All tests were performed on DNS-OARC's testbed.[1] The testbed consists of 16 high-specification servers on two dedicated and isolated 1000BASE-T Ethernet LANs. Some tests were performed on a single server while others involved the use of multiple servers. The operating systems used were CentOS 5.3 and FreeBSD 7.1. The name server implementations used were BIND 9.6.0-P1 and NSD 3.2.1.

1.1 Hardware

The DNS-OARC testbed is a collection of server and networking hardware used for simulations and performance testing. The testbed includes:

- (16) HP ProLiant DL140 G3 servers
- (2) D-Link DGS-3024 24-port gigabit Ethernet (1000BASE-T) switches
- (1) Dell 100BASE-T switch for out-of-band management

[1]DNS-OARC gratefully acknowledges the National Science Foundation grant OCI-0427144, a joint project between the Cooperative Association for Internet Data Analysis (CAIDA) and Internet Systems Consortium (ISC), for supporting the purchase of the testbed hardware.

- (1) Dell PowerConnect 3448 server configured as a PXE-boot server, management host, and proxy for the testbed servers

Each HP ProLiant DL140 G3 server has:

- (2) 3 GHz Xeon dual-core processors (four cores total)
- 16 GB RAM (one server has 32 GB RAM)
- (1) 70 GB SAS disk
- (2) Broadcom BCM5721 gigabit Ethernet adapters
- (1) HP Lights-Out 100i remote management port

1.2 Software

The following software was used for carrying out the tests in this analysis:

- BIND version 9.6.0-P1 (January 2009), `http://oldwww.isc.org/sw/bind/view/?release=9.6.0-P1`
- NSD version 3.2.1 (January 2009), `http://www.nlnetlabs.nl/projects/nsd/`
- CentOS version 5.3 (x86_64 arch) (April 2009), `http://wiki.centos.org/Manuals/ReleaseNotes/CentOS5.3`
- FreeBSD version 7.1-RELEASE (January 2009), `http://www.freebsd.org/releases/7.1R/announce.html`
- dnsperf version 1.0.1.0 (January 2008), `http://www.nominum.com/services/measurement_tools.php`
- tcpreplay version 3.4.1 (February 2009), `http://tcpreplay.synfin.net/trac/`
- NIST Net version 2.0.12c (final release) (July 2005), `http://www.antd.nist.gov/nistnet/`

1.3 Network

The DL140 servers each have two NICs and one management NIC. The servers are organized into three separate private networks:

- 10.1.0.0/24 for out-of-band management via HP integrated Lights-Out interfaces
- 10.2.0.0/24 for data (e.g., queries)
- 10.3.0.0/24 for data (e.g., replies)

During testing, various overlay networks were used on this foundation.

1.4 Zone Files

Root zone files of varying sizes and characteristics were created for use in various test scenarios. Zone files were generated to approximate some of the characteristics that could be expected in a root zone with as many as 10 million TLDs. In particular, care was taken to create a realistic distribution of domain name lengths and resource records.

Some of the assumptions lean towards the conservative. For example, the TLDs in the test zones have an average of six name servers each, mirroring the composition of the current root zone. However, if the root zone is opened to widespread registration, the number of name servers per domain may well be closer to the two or three generally typically seen for second-level domains.

Another assumption is the continuation of the use of "wide glue" [2] where for each NS resource record in the zone there is a corresponding glue record (or two, if IPv6 glue is added). Note that the nameservers in our root zones have no more than one A and/or AAAA record each. In other words, we don't have any multihomed nameservers with, for example, more than one A record. Furthermore, for new TLDs added to the existing root zone, we did not implement "nameserver sharing." Stated another way, new nameservers in the root zone are authoritative for only one TLD.

The following zone types were used for testing:

- U-4-DS0 :: Unsigned, primarily IPv4 glue,[2] no DS RRs
- U-6-DS0 :: Unsigned, IPv4 and IPv6 glue, no DS RRs
- S-6-DS10 :: DNSSEC-signed, IPv4 and IPv6 glue, DS RRs for 10% of TLDs
- S-6-DS50 :: DNSSEC-signed, IPv4 and IPv6 glue, DS RRs for 50% of TLDs
- S-6-DS100 :: DNSSEC-signed, IPv4 and IPv6 glue, DS RRs for 100% of TLDs

Note that in zones full IPv6 glue, each name server has both one A and one AAAA resource record associated with it; that is, there are no IPv6-only name servers.

Each zone type was generated in the following sizes:

- 1000 TLDs
- 10,000 TLDs
- 100,000 TLDs
- 1,000,000 TLDs
- 10,000,000 TLDs

For the DNSSEC-signed zones, the following DNSSEC parameters were used:

- Key-signing keys (KSKs) with a 2048-bit modulus and an exponent of 65537
- Zone-signing keys (ZSKs) with a 1024-bit modulus and an exponent of 65537
- DNSSEC algorithm 5 (RSA/SHA-1)

[2]Rather than remove the existing AAAA glue from the current root zone, we chose to maintain the existing ratio of AAAA-to-A glue records.

```
ALONG2.                         NS      F.NS.IFCOSTSRISKMEFIXQ5.NET.
                                NS      NS1.ALONG2.
                                NS      NS1.4FEETPRACTICESKYWAYS.ORG.
                                NS      NS2.ALONG2.
                                NS      NS2.LORD-2CLEARMR.COM.
                                NS      NS3.SCIENTISTARRANGE.COM.
                                NS      NS4.PRODUCTSART.ORG.
                                NS      DNS5.BOXSAMSTBURNOHBEDDAY.NET.
$TTL 3600       ; 1 hour
                                DS      23970 5 1 (
                                        3276067844B4FEC24606799FC7CFFAD23DB7
                                        AAC4 )
                                DS      34221 5 1 (
                                        DA2B2BE73C17B91DFAF2C2D9BF60F3D4D546
                                        FA61 )
$TTL 172800     ; 2 days
NS1.ALONG2.                     A       74.0.76.250
                                AAAA    2001:838:37:2::ac
NS2.ALONG2.                     A       209.104.216.198
                                AAAA    2001:8b0:d2:4::53
```

Figure 1.1: A few lines of a sample root zone file used in these tests.

- RRSIG validity period of seven days

Table 1.1 summarizes the key characteristics of the test zones, including the counts of NS, A, AAAA, and DS records for each zone, as well as the sizes of the zone file on disk. Note that zone files were created with the *libdns* library routines from the BIND9 source code. Figure 1.1 shows a few lines of a generated zone file.

	1K	10K	100K	1M	10M
U-4-DS0					
NS RRs	5,577	55,222	551,910	5,522,141	55,208,694
A RRs	5,114	54,759	551,447	5,521,678	55,208,231
AAAA RRs	233	727	5,873	55,114	552,567
DS RRs	0	0	0	0	0
File Size	373,194	3,975,009	40,106,983	403,750,677	4,063,616,250
U-6-DS0					
NS RRs	5,564	55,121	552,350	5,524,203	55,213,941
A RRs	5,101	54,658	551,887	5,523,740	55,213,478
AAAA RRs	4,209	53,766	550,995	5,522,848	55,212,586
DS RRs	0	0	0	0	0
File Size	478,039	5,391,044	54,812,839	550,932,393	5,533,414,033
S-6-DS10					
NS RRs	5,521	55,425	552,103	5,520,185	55,213,776
A RRs	5,058	54,962	551,640	5,519,722	55,213,313
AAAA RRs	4,166	54,070	550,748	5,518,830	55,212,421
DS RRs	172	1,950	20,036	199,636	2,001,164
File Size	491,582	5,622,517	56,879,721	571,450,628	5,743,522,564
S-6-DS50					
NS RRs	5,464	55,079	551,800	5,524,164	55,214,449
A RRs	5,001	54,616	551,337	5,523,701	55,213,986
AAAA RRs	4,109	53,724	550,445	5,522,809	55,213,093
DS RRs	728	9,766	99,654	999,946	9,996,044
File Size	543,745	6,425,989	65,206,420	655,762,117	6,581,421,523
S-6-DS100					
NS RRs	5,510	55,519	552,447	5,524,835	55,214,832
A RRs	5,047	55,056	551,984	5,524,372	55,214,369
AAAA RRs	4,155	54,164	551,092	5,523,480	55,213,477
DS RRs	1,440	19,440	199,440	1,999,440	19,999,440
File Size	624,791	7,471,784	75,739,257	760,623,417	7,629,876,108

Table 1.1: RR counts and on-disk file sizes (bytes) of zones used for this study. Note that the *U-4-DS0* zones have some AAAA records to match the ratio of AAAA-to-A RRs in the current root zone.

Chapter 2

Impact on Zone Size

This task examines the impact of an increased number of TLDs on the size of the root zone. We were asked to consider the following:

> *How will the addition of new TLDs increase zone size. Key variables to examine are:*
>
> a. *Addition of IPv6 addresses for all name servers in the root zone*
> b. *Whether the zone is DNSSEC-signed (using anticipated root DNSSEC keys and key sizes)*
> c. *Percentage of TLDs have DS RRs with at least 10%, 50%, and 100% considered*

The size of the zone file may be considered in several different ways. Typically it is taken to be the size of the file itself, i.e., the text file in the format described in Section 5.1 of RFC 1035. However, from an operational perspective, the most significant measure of zone size is the amount of memory required by a name server to load and serve it. The following analysis focuses on this metric.

	Zone Size (#TLDs)				
Zone Type	1K	10K	100K	1M	10M
U-4-DS0	0.04	0.05	0.19	1.52	14.60
U-6-DS0	0.04	0.06	0.26	2.21	21.48
S-6-DS10	0.04	0.07	0.30	2.64	26.27
S-6-DS50	0.04	0.07	0.32	2.81	28.13
S-6-DS100	0.04	0.07	0.34	3.02	29.73

Table 2.1: Memory usage (in gigabytes) for different zone types and sizes – BIND.

2.1 Setup

BIND and NSD were installed and configured on a testbed host with 32 GB RAM. A test harness written in Perl automatically cycles through each test zone, launches the name server and then measures and records the amount of memory used by the name server instance. A zone is considered fully loaded only after the name server correctly answers a query for a resource record at the end of the zone.

We use the `pmap` utility to measure process memory usage. Note that these measurements include all memory used by the process, including that used by shared libraries. `pmap` is available for Linux as part of the `procps` package[3] and for FreeBSD as a port[4]. Figure 2.1 shows an example of the output of pmap.

2.2 BIND Results

The raw data for BIND is shown in Table 2.1. The given numbers represent the process size measured in gigabytes. We chose to show all numbers with the same units to make them easy to visually compare.

Figure 2.2 shows the BIND results graphically. Note that both the x- and y-axes are scaled logarithmically to render the results most clearly.

Figure 2.3 shows the relative change in memory usage compared to the U-

```
# pmap -x 6235
6235:   /usr/sbin/named -u bind
Address           Kbytes     RSS    Anon  Locked Mode   Mapping
0000000000400000     380       -       -       - r-x--  named
000000000065e000      24       -       -       - rw---  named
0000000000664000       4       -       -       - rw---    [ anon ]
0000000000863000      24       -       -       - rw---  named
000000000893f000 3012920       -       -       - rw---    [ anon ]
00000032be800000     104       -       -       - r-x--  ld-2.5.so
00000032bea1a000       4       -       -       - r----  ld-2.5.so
00000032bea1b000       4       -       -       - rw---  ld-2.5.so
00000032bec00000    1320       -       -       - r-x--  libc-2.5.so
00000032bed4a000    2048       -       -       - -----  libc-2.5.so
00000032bef4a000      16       -       -       - r----  libc-2.5.so
00000032bef4e000       4       -       -       - rw---  libc-2.5.so
00000032bef4f000      20       -       -       - rw---    [ anon ]
00000032bf000000       8       -       -       - r-x--  libdl-2.5.so
00000032bf002000    2048       -       -       - -----  libdl-2.5.so
00000032bf202000       4       -       -       - r----  libdl-2.5.so
00000032bf203000       4       -       -       - rw---  libdl-2.5.so
00000032bf400000      84       -       -       - r-x--  libnsl-2.5.so
00000032bf415000    2044       -       -       - -----  libnsl-2.5.so
00000032bf614000       4       -       -       - r----  libnsl-2.5.so
00000032bf615000       4       -       -       - rw---  libnsl-2.5.so
00000032bf616000       8       -       -       - rw---    [ anon ]
00000032bf800000     292       -       -       - r-x--  libisc.so.50.0.2
00000032bf849000    2044       -       -       - -----  libisc.so.50.0.2
00000032bfa48000       8       -       -       - rw---  libisc.so.50.0.2
00000032bfc00000    1372       -       -       - r-x--  libdns.so.50.0.3
00000032bfd57000    2044       -       -       - -----  libdns.so.50.0.3
00000032bff56000      28       -       -       - rw---  libdns.so.50.0.3
00000032c0800000      80       -       -       - r-x--  libz.so.1.2.3
00000032c0814000    2044       -       -       - -----  libz.so.1.2.3
00000032c0a13000       4       -       -       - rw---  libz.so.1.2.3
00000032c1000000      64       -       -       - r-x--  liblwres.so.50.0.0
00000032c1010000    2048       -       -       - -----  liblwres.so.50.0.0
00000032c1210000       4       -       -       - rw---  liblwres.so.50.0.0
00000032c1400000    1172       -       -       - r-x--  libcrypto.so.0.9.8b
00000032c1525000    2048       -       -       - -----  libcrypto.so.0.9.8b
00000032c1725000     124       -       -       - rw---  libcrypto.so.0.9.8b
00000032c1744000      16       -       -       - rw---    [ anon ]
0000003ac1e00000      36       -       -       - r-x--  libbind9.so.50.0.1
0000003ac1e09000    2048       -       -       - -----  libbind9.so.50.0.1
0000003ac2009000       4       -       -       - rw---  libbind9.so.50.0.1
0000003ac2200000      28       -       -       - r-x--  libisccc.so.50.0.0
0000003ac2207000    2044       -       -       - -----  libisccc.so.50.0.0
0000003ac2406000       4       -       -       - rw---  libisccc.so.50.0.0
0000003ac2600000      92       -       -       - r-x--  libisccfg.so.50.0.0
0000003ac2617000    2048       -       -       - -----  libisccfg.so.50.0.0
0000003ac2817000      24       -       -       - rw---  libisccfg.so.50.0.0
00002b86043ce000       4       -       -       - rw---    [ anon ]
00002b86043d3000      24       -       -       - rw---    [ anon ]
00002b86043d9000      40       -       -       - r-x--  libnss_files-2.5.so
00002b86043e3000    2044       -       -       - -----  libnss_files-2.5.so
00002b86045e2000       4       -       -       - r----  libnss_files-2.5.so
00002b86045e3000       4       -       -       - rw---  libnss_files-2.5.so
00002b86045e4000    1752       -       -       - rw---    [ anon ]
00002b86047bb000   27820       -       -       - rw---    [ anon ]
00002b8606342000   55900       -       -       - rw---    [ anon ]
00002b860a7bd000   29528       -       -       - rw---    [ anon ]
00007fffa66c6000      88       -       -       - rw---    [ stack ]
ffffffffff600000    8192       -       -       - -----    [ anon ]
----------------  ------  ------  ------  ------
total kB         3166200       -       -       -
```

Figure 2.1: Example output of pmap for an instance of BIND.

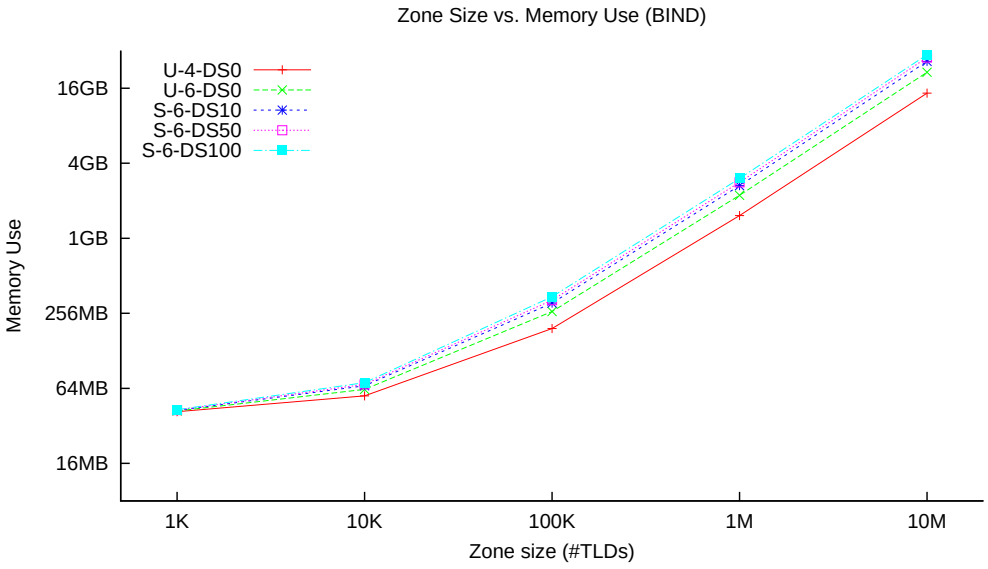

Figure 2.2: Zone size vs. memory use – BIND.

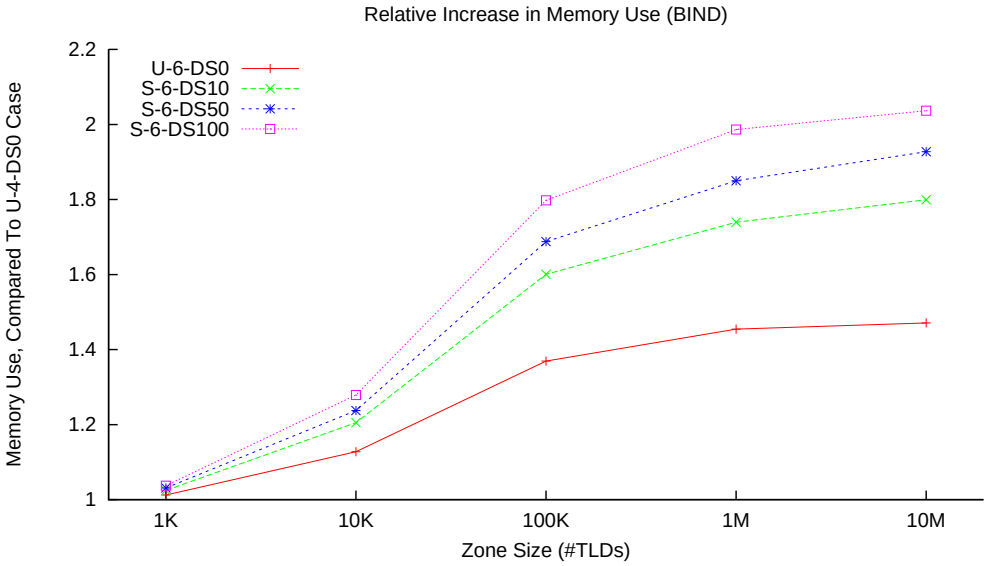

Figure 2.3: Relative change in memory usage compared to the U-4-DS0 case – BIND.

	Zone Size (#TLDs)				
Zone Type	1K	10K	100K	1M	10M
U-4-DS0	0.02	0.04	0.21	1.92	18.73
U-6-DS0	0.02	0.05	0.26	2.42	23.74
S-6-DS10	0.02	0.05	0.31	2.90	N/A
S-6-DS50	0.02	0.05	0.33	3.11	N/A
S-6-DS100	0.02	0.06	0.36	3.38	N/A

Table 2.2: NSD memory usage (in gigabytes) for different zone types and sizes. Note that NSD was unable to load the signed 10 million TLD zone on this hardware (32 GB RAM).

4-DS0 case. For example, you can see that for a 1 million TLD zone file, adding both full IPv6 glue and full DS records doubles the memory usage.

2.3 NSD Results

The raw data for NSD is shown in Table 2.2. The given numbers represent the process size, measured in gigabytes. Note that NSD was unable to load the signed 10 million TLD zones on this platform, which had 32 GB of RAM. Compared to BIND, NSD uses less memory for the 1K and 10K TLD zones, about the same for the 100K TLD zone, and more memory for the 1M and larger zones.

Figure 2.4 shows the NSD results graphically. Note that both the x- and y-axes are scaled logarithmically.

Figure 2.5 shows the relative change in memory usage compared to the U-4-DS0 case.

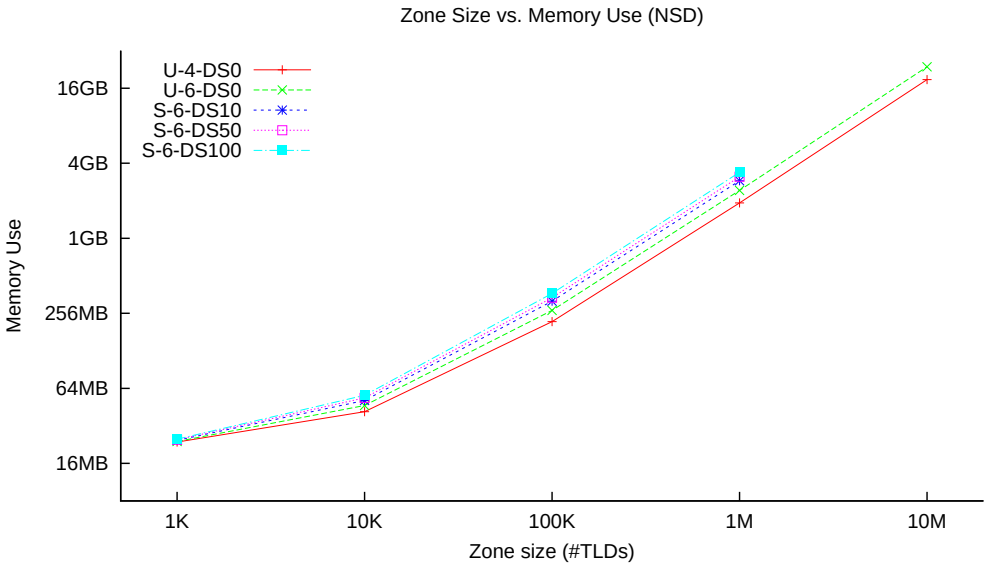

Figure 2.4: Zone size vs. memory use – NSD.

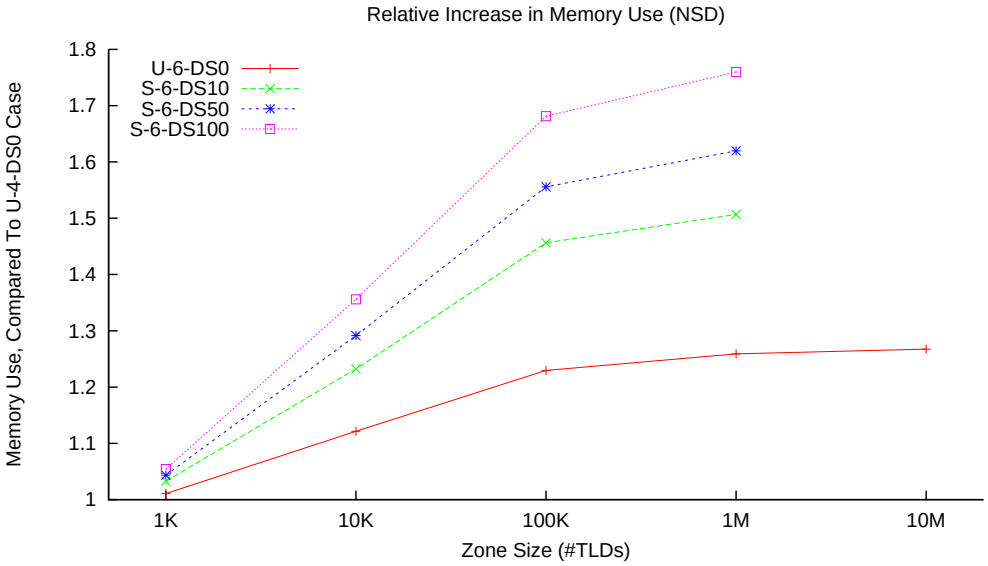

Figure 2.5: Relative change in memory usage compared to the U-4-DS0 case – NSD.

Chapter 3

Impact on Response Latency

This task examines the impact of an increased number of TLDs on the latency of responses to DNS queries. We were asked to consider the following:

> *Determine by way of experimentation the impact in terms of response latency and server load on an L root server analog as a function of zone size. Key variables to examine are:*
>
> a. *Addition of IPv6 addresses for all name servers in the root zone*
> b. *Whether the zone is DNSSEC-signed (using the anticipated DNSSEC keys and key sizes)*
> c. *Percentage of TLDs have DS RRs, with at least 10%, 50%, and 100% considered.*
> d. *Ratio of queries for existent/non-existent top-level domains, with (at least) 10 (existent) : 1 (non-existent), 1:1, 1:10, 1:100, and 1:1000.*

3.1 Setup

For testing latency we used the setup shown in Figure 3.1. NSD 3.2.1 and BIND 9.6.0-P1 were installed on the name server using the compilation op-

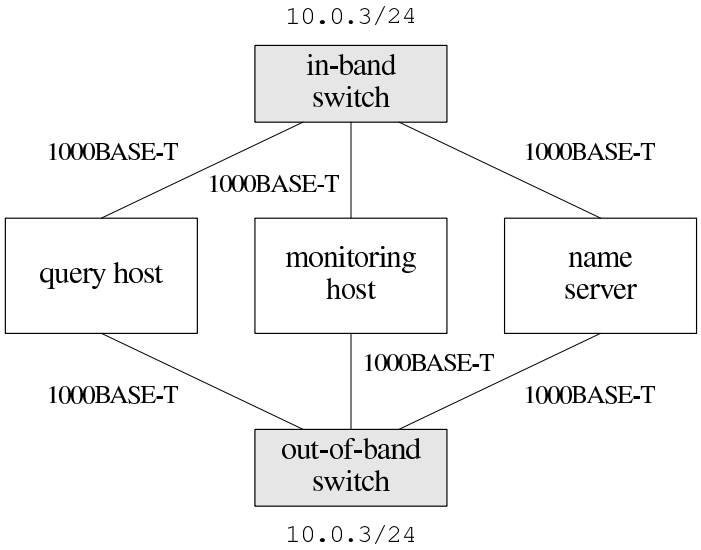

Figure 3.1: Setup for measuring query latency.

tions and configurations detailed in Appendix A. We used the testbed host with 32 GB RAM as the name server. Tcpreplay was installed on the query host. Pcap files of queries of varying types were generated using `querygen`, a utility we wrote, and moved to the query host. The switch was configured to mirror in-band traffic between the name server and the query server to the monitoring host. Tcpdump was used to capture traffic from the mirrored port. All management (e.g., SSH sessions for running and monitoring the setup) was over the out-of-band network, effectively isolating the DNS traffic to be measured.

The `querygen` utility can build query streams with various characteristics. Some of the available parameters are:

- Start time, end time, and/or duration of the query stream
- Percentage of queries with EDNS0
- Percentage of queries with DO-bit set
- Percentage of queries with bad UDP checksums
- Percentage of queries without UDP checksums

- Percentage of queries with the DF ("don't fragment") bit set
- Source IP address distribution
- Source port distribution
- Query type (QTYPE) distribution
- Query interval distribution (i.e., the distribution of time intervals between successive queries)
- Lists of names to use for QNAMEs

The distribution parameters are files containing key-value pairs representing the probability distribution of various things. For example, the source address distribution file is a list of source addresses and their relative (or absolute) frequencies. `querygen` then builds query streams with characteristics that match the input parameters by sampling from the input distributions using probability distribution functions.

Unless otherwise noted, the query streams used for latency testing had the following characteristics:

- The query streams were 60 seconds in duration
- The query characteristics were taken from queries to the L-root LAX node captured during the 2009 DITL Data Collection Project[6]:
  - The query rate was 5000 queries per second[1]
  - 75% of queries had EDNS0 records
  - 70% of queries had the DO-bit set
  - 77% of queries had the DF ("don't fragment") bit set
  - 0.08% of queries had no UDP checksums
  - 0.01% of queries had bad UDP checksums

[1]Certainly a DNS root server sees more than 5,000 queries per second in aggregate. We believe, however, that most DNS root servers are highly distributed, either via global anycast or local load balancers, so that any single server/instance/node typically sees no more than 5,000 queries per second.

– 30% of queries were for non-existent TLDs

Other characteristics were varied as needed to elicit differences in query latency, and are noted in each case.

The test setup had some simplifications that could have served to improve latency slightly:

- TCP-based DNS activity was not simulated
- Damaged packets were not simulated
- The test network was free of any competing (non-DNS) traffic

3.2 Results

3.2.1 Effect of Zone Size on Latency

Increased zone size has the potential to affect latency. At a minimum, there is a higher likelihood of hash collisions or increased tree depth in the data structures employed by the name server, resulting is additional processing for some queries. If the zone is sufficiently large, the free list drops to a point where the system commits resources to conserve memory, imposing additional load and further delaying replies.

We surveyed the relationship between zone size and reply latency. Queries were sent at a rate of 5000 queries per second to instances of BIND and NSD serving both unsigned and signed zones of varying sizes.

Table 3.1 shows the median latencies for all tested configurations. In most cases the measured latency is less than 1 millisecond. BIND, however, seems to labor when serving larger signed zones. For the 100K TLD zone, we observed a median latency of 77 milliseconds. With the larger zones, more than half of the transactions took longer than 4 seconds. Table 3.2 shows the percentage of transactions completing in less than 4 seconds for all tested configurations.

Figure 3.2 shows the observed latencies for BIND serving five different sizes of unsigned root zones. The smallest zone contains 1000 TLDs; each successive

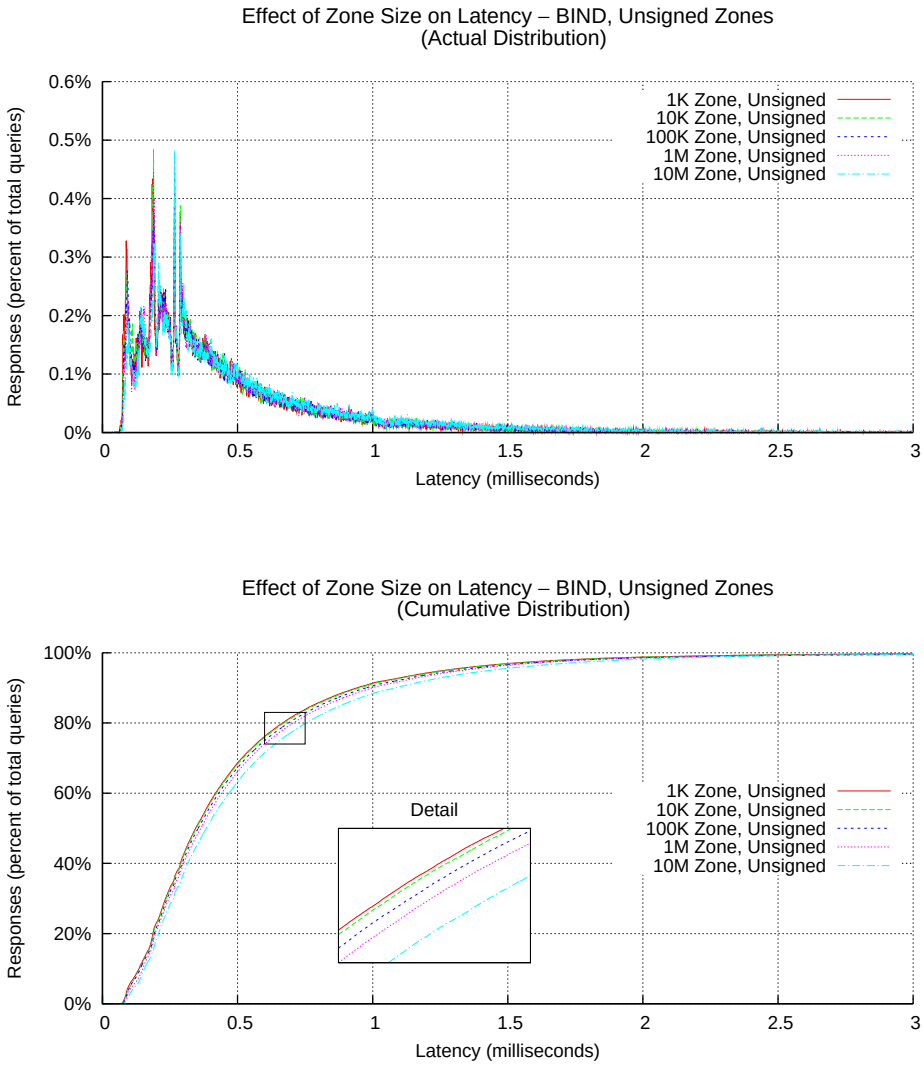

Figure 3.2: Effect of zone size on latency – BIND, unsigned zones.

Software	Zone Type	Zone Size				
		1K	10K	100K	1M	10M
BIND	unsigned	0.3	0.3	0.4	0.4	0.4
BIND	signed	0.4	0.4	77.7	>4000	>4000
NSD	unsigned	0.1	0.1	0.1	0.1	0.1
NSD	signed	0.1	0.1	0.1	0.1	0.3†

Table 3.1: Median latencies (milliseconds) for different zone types and sizes on BIND and NSD. †Note the NSD signed measurement is with a 4.5M TLD zone.

Software	Zone Type	Zone Size				
		1K	10K	100K	1M	10M
BIND	unsigned	100	100	100	100	100
BIND	signed	100	100	71	19	11
NSD	unsigned	100	100	100	100	100
NSD	signed	100	100	100	97	59†

Table 3.2: Percent of queries completed within 4 seconds on BIND and NSD. †Note the NSD signed measurement is with a 4.5M TLD zone.

zone is a factor of 10 larger than the previous one. The first plot shows the actual distribution of queries, i.e., a histogram of the observed latencies. The second plot shows the cumulative distribution.

The data shows that there is a minor but noticeable effect with an increase in zone size. The inset shows that the difference is most noticeable for the largest (10 million TLD) zone, indicated by the light blue line. The difference in latency between the smallest and largest zone is no more than a tenth of millisecond on average. The distribution of queries is markedly similar for all zone sizes, though the tail becomes longer as the zone file becomes larger.

The distributions for all five zones show several notable peaks near the 0.1, 0.2, and 0.3 millisecond mark, almost as if there were an underlying harmonic at work. These peaks may coincide with some internal queuing or interrupt-driven mechanism within BIND.

Figure 3.3 shows the observed latencies for NSD serving the same five zones. NSD and BIND have noticeably different latency distributions. NSD serves nearly 100% of the queries in 0.3 milliseconds or less, whereas BIND serves

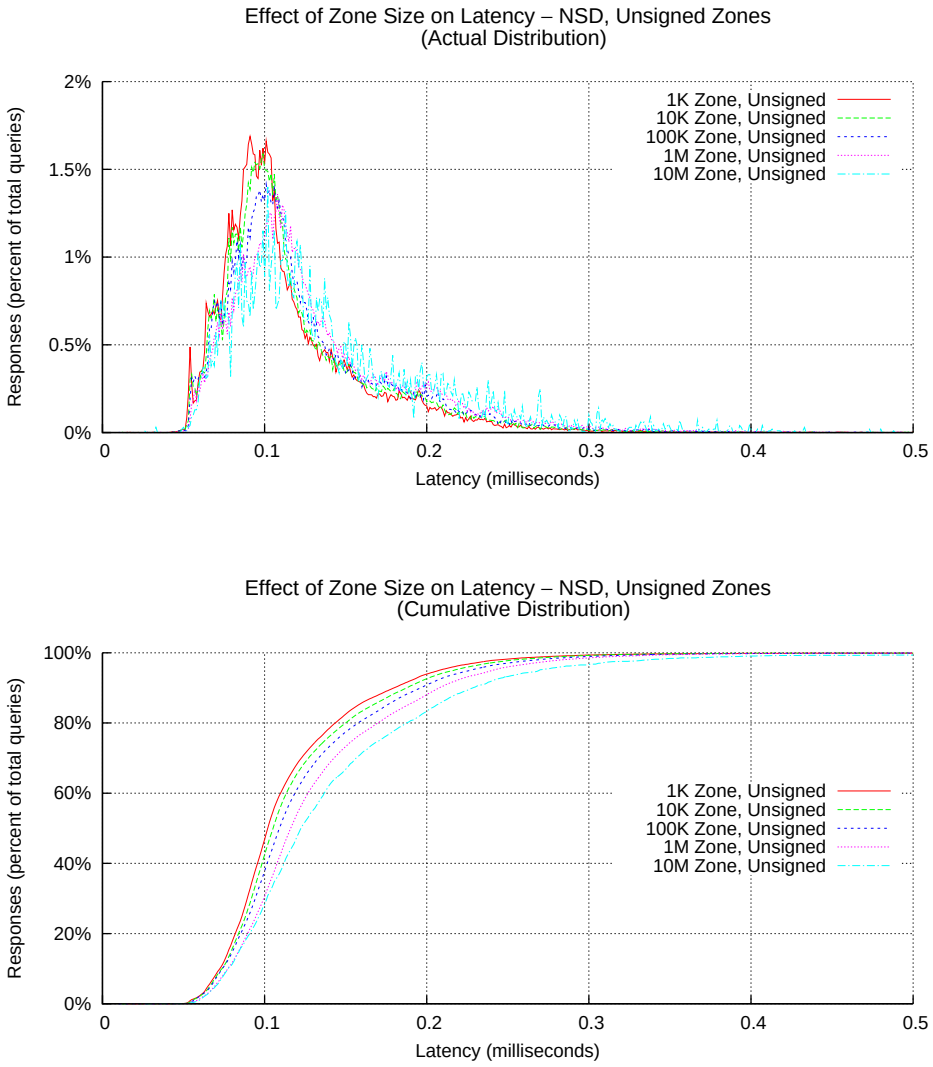

Figure 3.3: Effect of zone size on latency – NSD, unsigned zones.

fewer than 45% at that level for even the smallest zone. As with BIND, NSD exhibits some reduction in performance with larger zone sizes, but again this reduction is minute.

It is worth noting that for both BIND and NSD, the increase in latency is *orders of magnitude smaller* than the latencies of the network links between the root servers and even centrally-positioned resolvers. The increases shown here would be far below the threshold of perception of any end user, and it is difficult to envisage any scenario where they would have an impact on any Internet-connected device or application.

Figure 3.4 shows the observed latencies for BIND serving the signed versions of the most fully populated zones, i.e., the ones with AAAA glue for all name servers and with two DS resource records for each TLD. A dramatic reduction in performance is evident with increasing zone size. Note that this plot has been rendered with a logarithmic time scale to make the changes in performance clearer. The distribution of latencies for the 1K signed zone file looks much like that seen for the unsigned zones, but differences become apparent with the 10K signed zone. The impact of the larger zones is extreme, with nearly 30% of queries lost for the 100K signed zone, and more than 80% for the largest zones.

While some of the increase may be attributable to the increase in number of size and number resource records in the signed zones, most of it is due to the additional overhead of DNSSEC: more than 70% of queries have the DO-bit set, and BIND has to do additional processing for these queries. In particular, if the query is for a non-existent domain, the server must retrieve the correct NSEC RRs and associated RRSIG RRs where previously only the SOA RR was needed. Note that, contrary to a common misperception, none of the overhead is cryptographic, e.g., requiring the generation of signatures or the validation of data. Signatures are generated at the time the zone is signed, and no validation is performed by an authoritative-only server.

Comparing the 100K signed zone with the 1M unsigned zone is instructive. The latter is larger by every metric: it is 2.8 times larger and has 5.1 times more resource records, and yet BIND exhibits much less latency and far fewer dropped queries when serving it.

The extreme reduction in performance seen for BIND serving the 10M signed zone should be kept in perspective: BIND starts to use swap when this zone

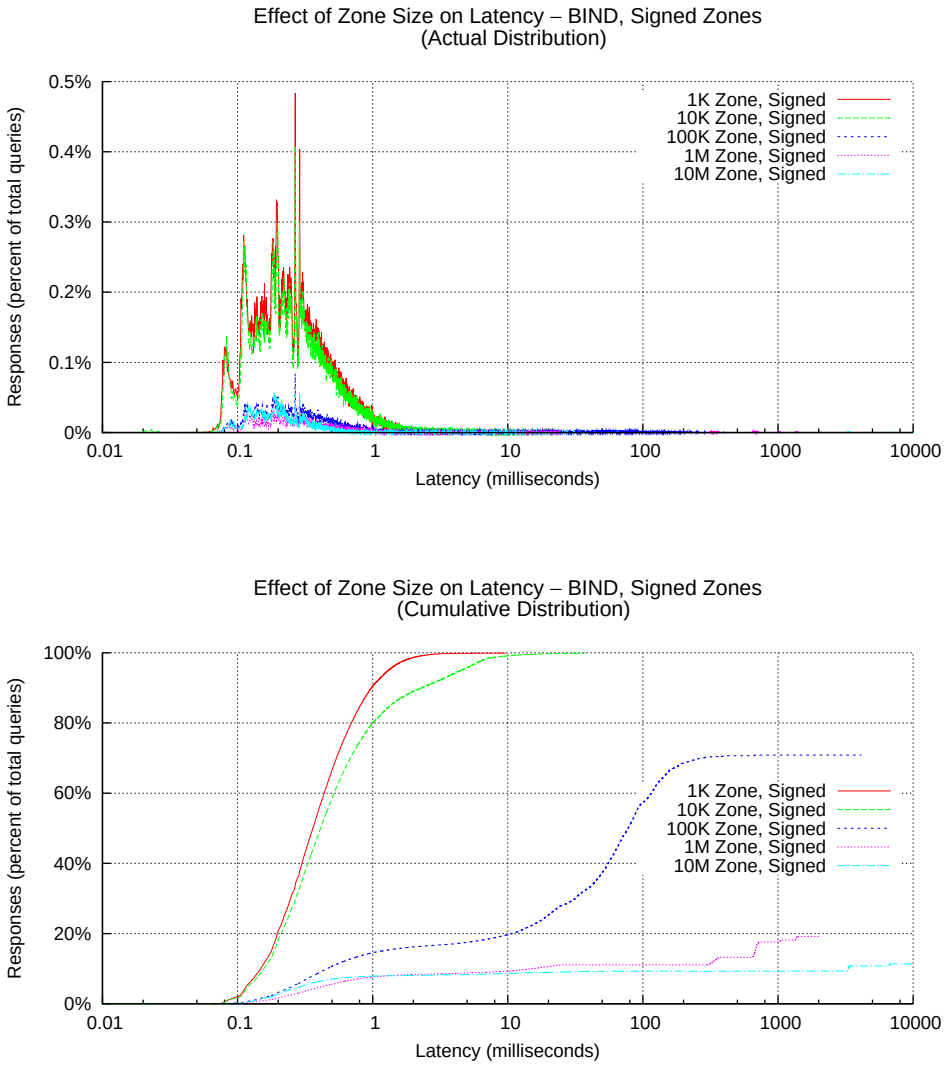

Figure 3.4: Effect of zone size on latency – BIND, signed zones.

is loaded on the testbed host with 32 GB RAM, so some of the degradation may be due to paging. This is not the case with the 100K and 1M zones.

It should be noted that DNSSEC was enabled in the configuration for both signed and unsigned zones, so that changes in latency are due to the presence of DNSSEC resource records alone, not to the configuration of BIND.

Figure 3.5 shows the observed latencies for NSD serving the same zones. In marked contrast to BIND, DNSSEC imposes very little overhead on NSD when serving signed zones. Remarkably, the distributions of latencies are comparable to those seen for the unsigned zones. However, NSD's performance does start to suffer when serving the 1M signed zone, where a little over 3% of the queries are not answered in 4 seconds or less.

As noted in the previous chapter, NSD cannot load the 10M signed zone, so instead a 4.5 million TLD signed zone – the largest that NSD can load in 32 GB of memory – has been substituted. Over 40% of queries are not answered within 4 seconds for this zone, demonstrating that NSD is not impervious to resource exhaustion.

We wanted to go back and take a closer look at BIND's performance characteristics when serving large signed zones. One variable we investigated was query rate. Figure 3.6 shows the results when the queries were replayed at 10% of the original rate, or 500 qps, and 1% of the original rate, or 50 qps. Even at these slower speeds, significant performance penalties are apparent. It is particularly noteworthy that with a 1M TLD zone and a rate of 500 qps, nearly 4% of queries still go unanswered (within 4 seconds).

We also wanted to rule out the possibility that the underlying operating system was playing a deciding role in performance. Figure 3.7 shows the results when the same trials shown in Figure 3.4 were repeated using FreeBSD 7.1 rather than CentOS 5.3 as the operating system.

There is some improvement in the results: the tail of the distribution is less attenuated, and the response rate is higher. Ultimately, though, the performance for the 100K and 1M zone falls short of what would be suitable for a production environment.

This section clearly showed that both NSD and BIND struggle to serve zones significantly larger than the signed 1M TLD zone on a host with 32 GB memory. No hosts were available with more than 32 GB, so we were unable to

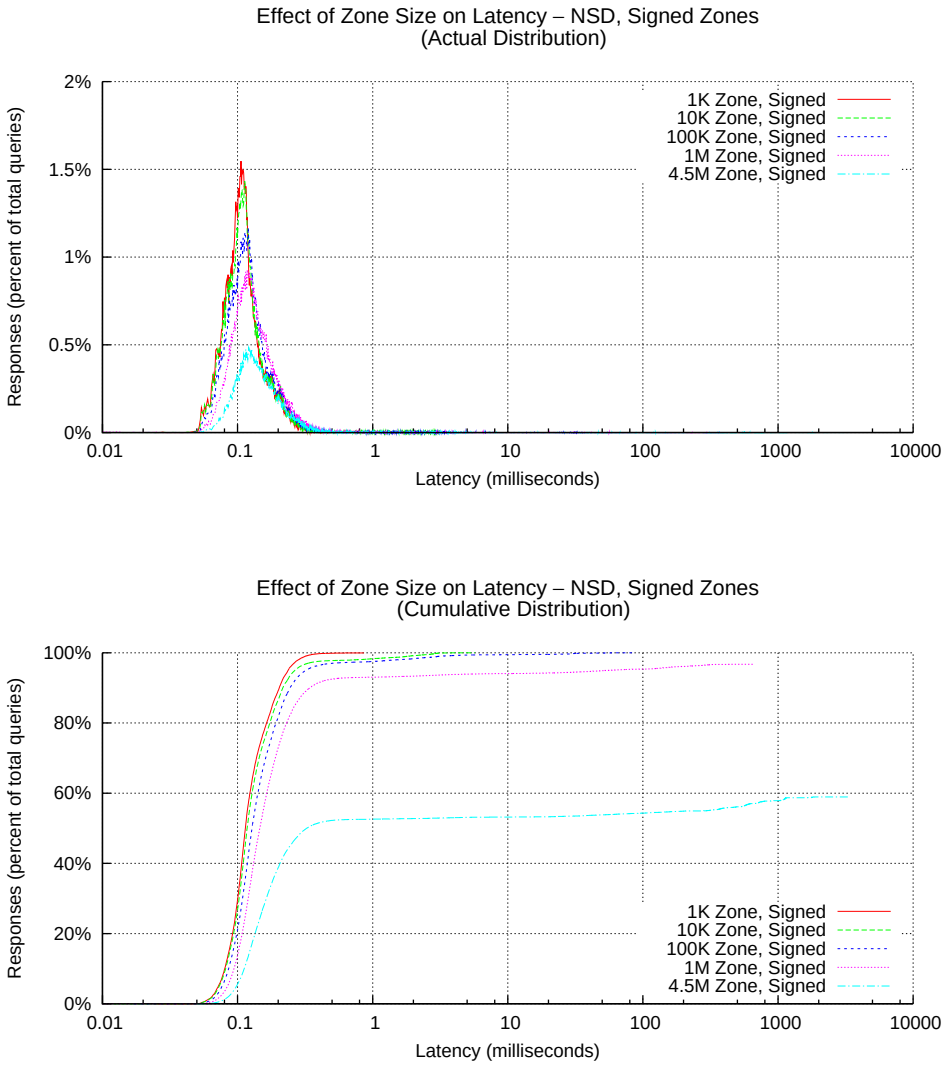

Figure 3.5: Effect of zone size on latency – NSD, signed zones.

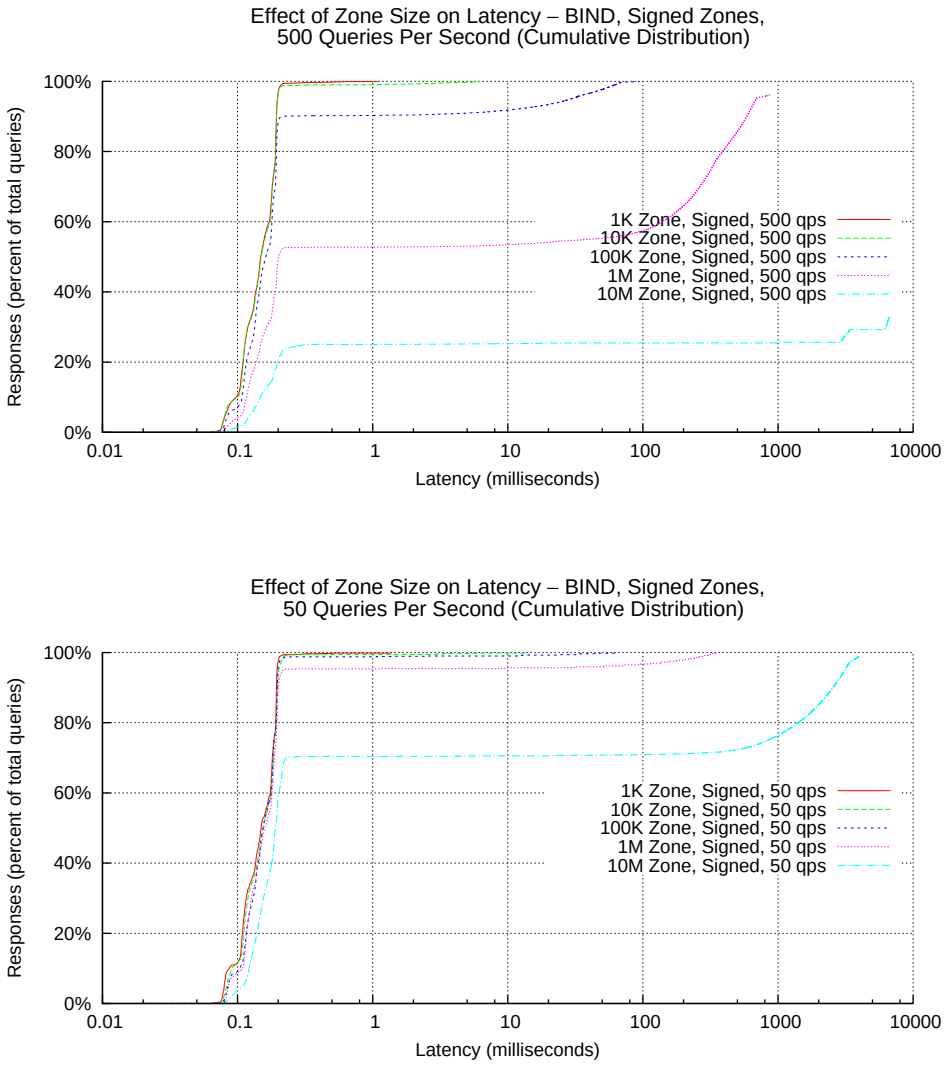

Figure 3.6: Effect of zone size on latency – BIND, signed zones, 500 and 50 queries per second.

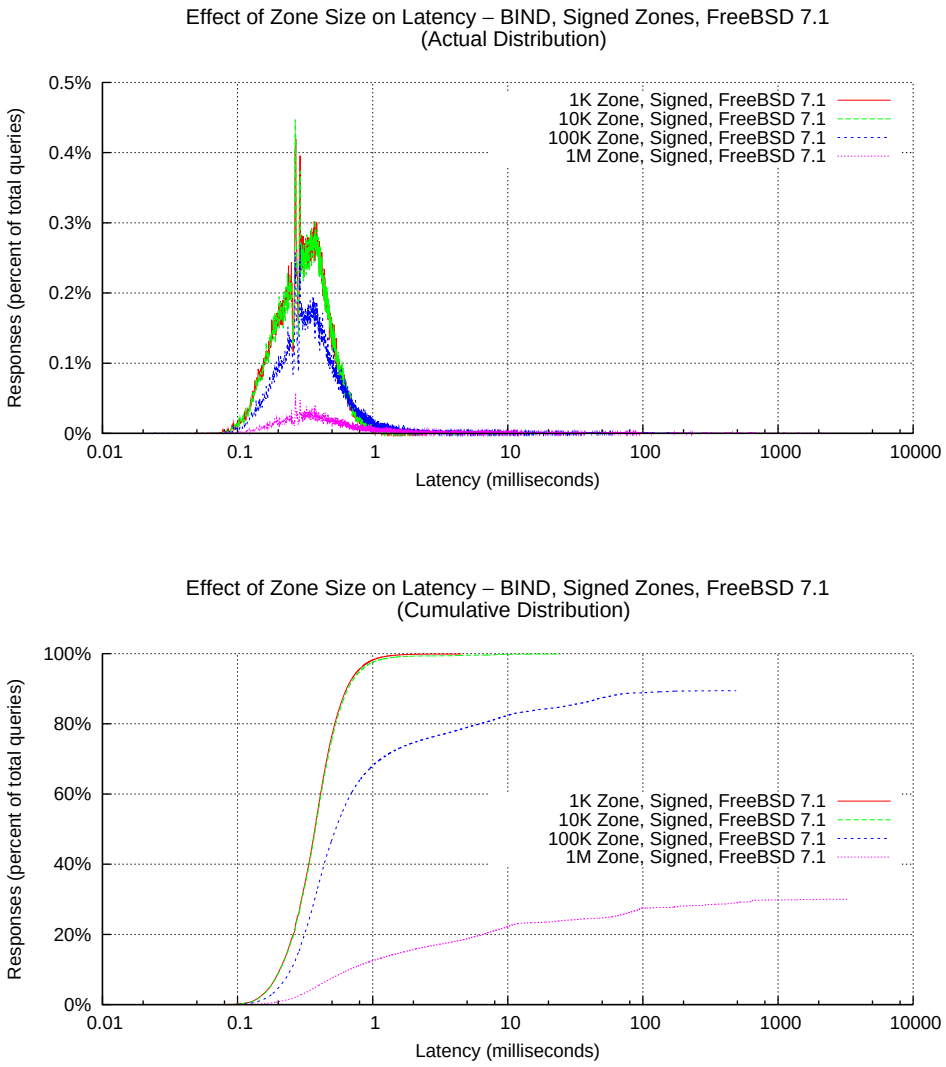

Figure 3.7: Effect of zone size on latency – BIND, signed zones, FreeBSD 7.1.

determine to what degree additional memory might alleviate the performance shortfall. Under the circumstances, continuing to test largest zones seemed counterproductive, so we omitted them from the remainder of our latency testing.

3.2.2 Latency Differences Between Zones with Sparse and Full IPv6 Glue

The addition of IPv6 glue records to the root zone has the potential to affect name server latency as referral replies become larger to accommodate the additional information, and some additional processing is needed to assemble and send the reply message. The logical outcome to full IPv6 deployment is IPv6 glue records for every name server listed in the root zone. We investigated the impact that a root zone fully populated with IPv6 glue would have on latency.

We replayed the query stream from the previous section to BIND and NSD serving the 1K TLD root zone with sparse IPv6 glue only, and then the same zone with full IPv6 glue. In keeping with ICANN's policy of using mandatory glue, this meant there was a AAAA RR as well as an A RR for every NS RR in the zone.

The mean size of DNS messages containing referral responses grew from 452 to 461 octets;[2] however this resulted in only a minimal change in latency. Figures 3.8 (BIND) and 3.9 (NSD) show the effect on latency.

[2]Note that the mean response size reflects the fact that a portion (22.7%) of the reply messages are limited to 512 octets, either because the query has no EDNS0, or advertises an EDNS0 payload size of 512. In this case, the server removes as many RRs from the Additional section as is necessary to keep the response size within 512 octets.

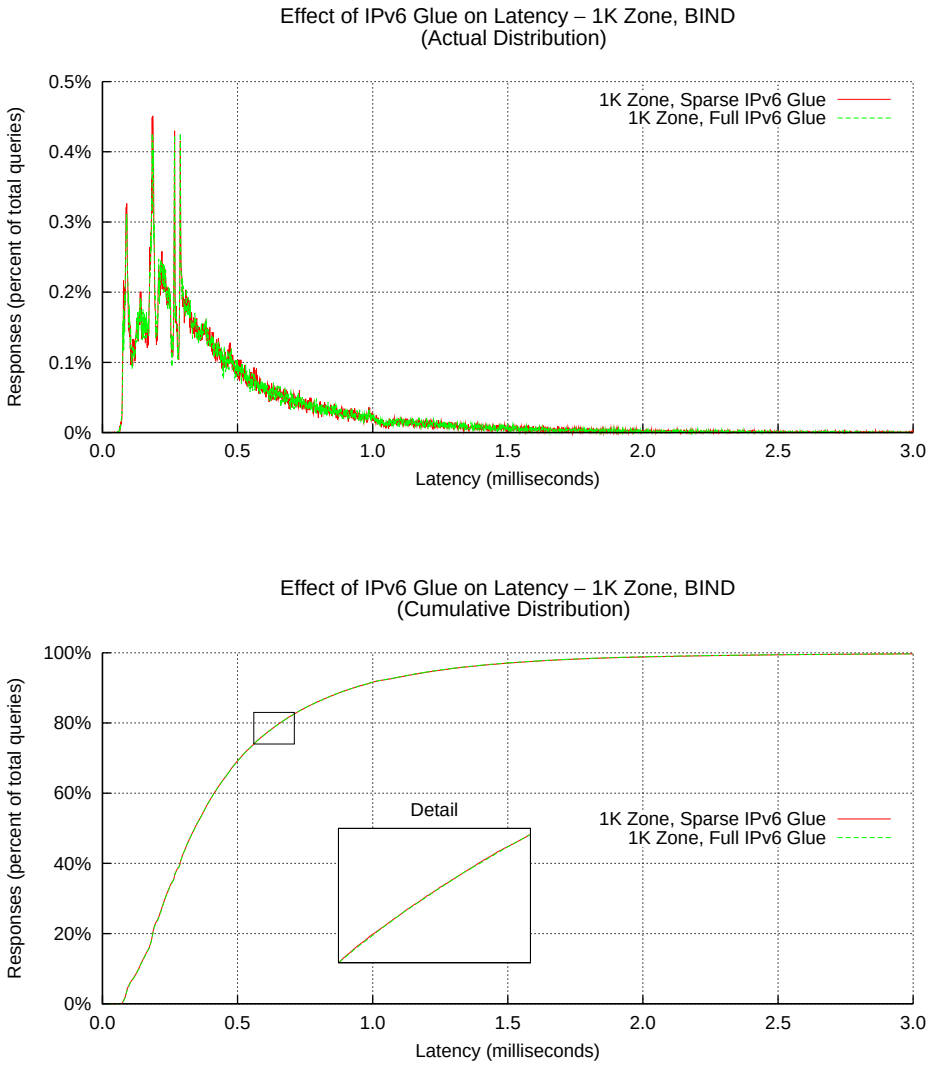

Figure 3.8: Effect of IPv6 glue on latency – 1K zone, BIND.

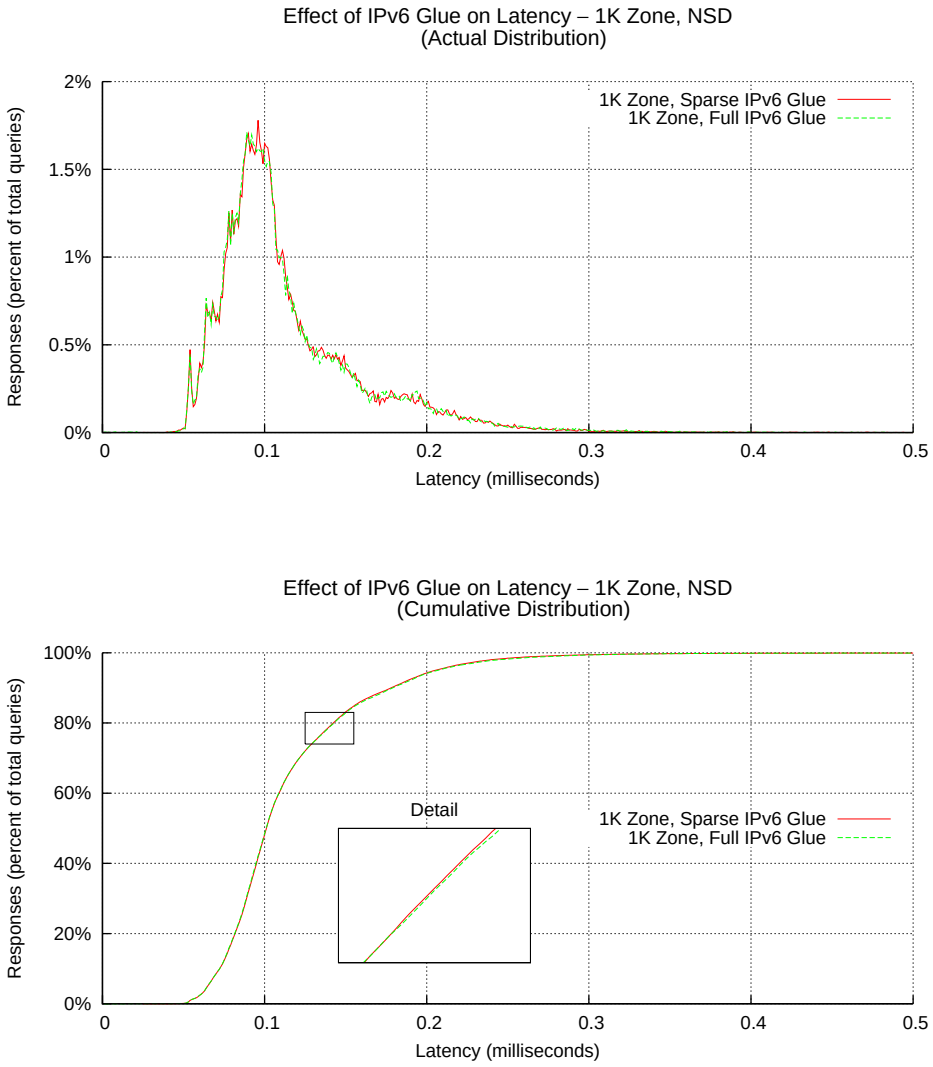

Figure 3.9: Effect of IPv6 glue on latency – 1K zone, NSD.

We then replayed the query stream to BIND and NSD serving the 1M TLD root zone with both sparse and full glue. This time the mean referral response size increased from 261 to 280 octets. Again, the increase in latency was small. Figures 3.10 and 3.11 show the differences.

We also tested intermediate (10K, 100K) zone sizes, and the results were comparable. Based on these results, it seems safe to assume that adding the maximum amount of IPv6 glue to the root zone, whatever the size, will have minimal marginal impact.

3.2.3 Latency Differences Between Unsigned and Signed Zones

During the examination of the impact of zone size on latency in Section 3.2.1, it was apparent that signing a zone had an impact on latency disproportionate to its larger size when compared with the unsigned counterpart. In this section we look more closely at these differences.

Figures 3.12 and 3.13 show the latency distributions for an unsigned and signed 1K TLD root zone file when the 5000 qps sample of queries to L-root server are replayed to it. There is some – though little – divergence in latency.

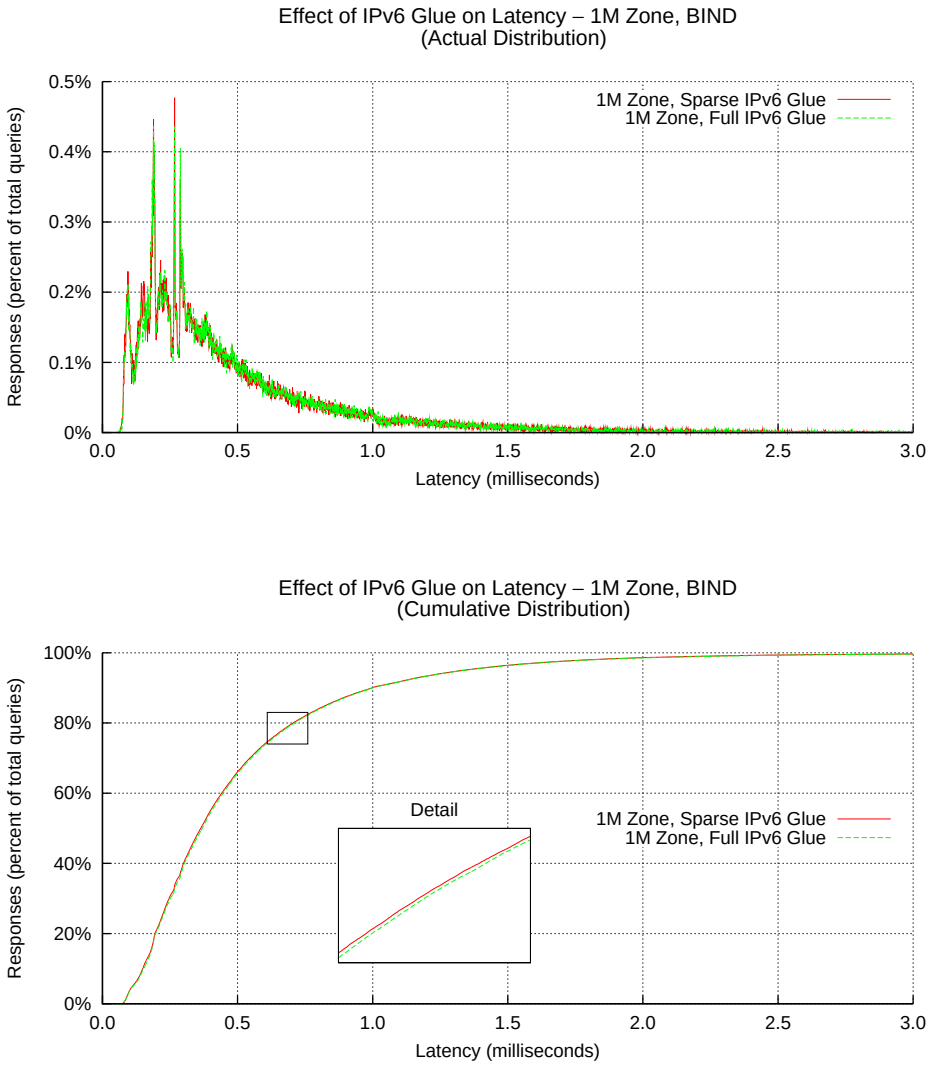

Figure 3.10: Effect of IPv6 glue on latency – BIND, 1M zone.

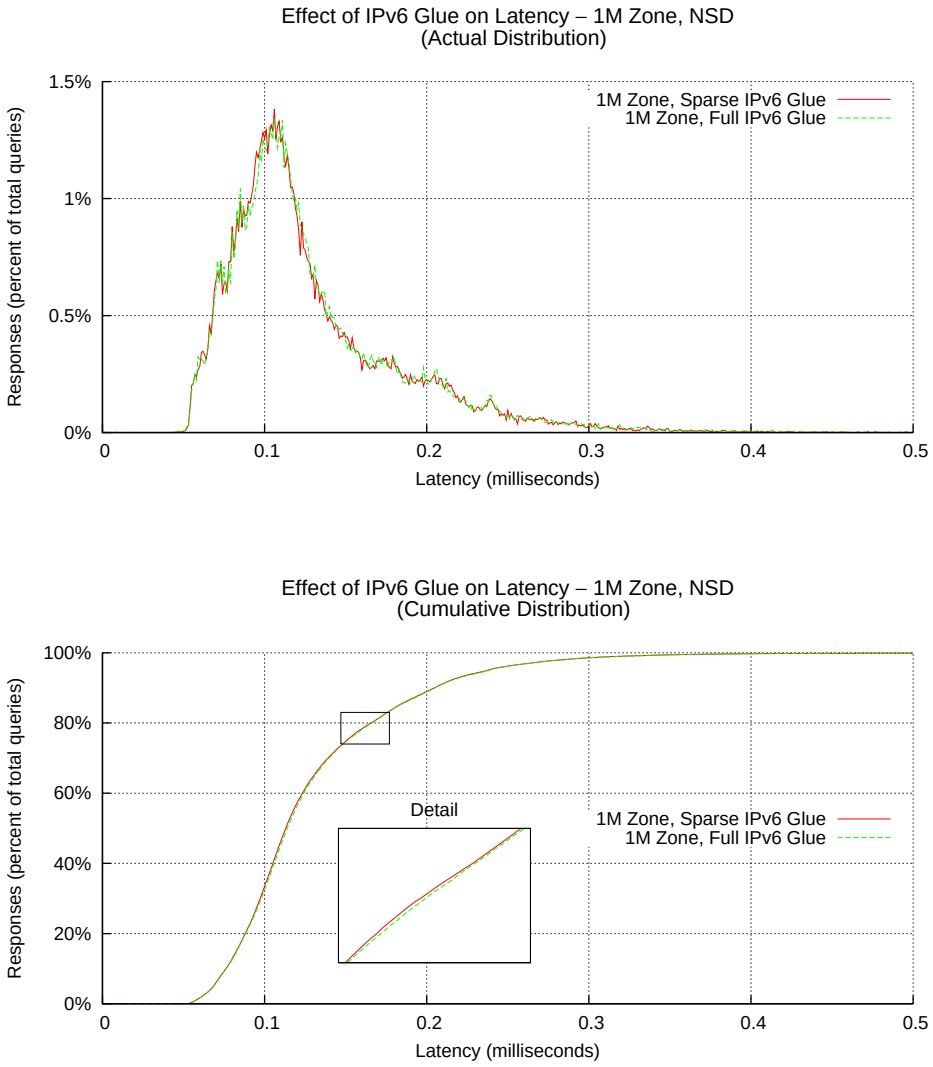

Figure 3.11: Effect of IPv6 glue on latency – NSD, 1M zone.

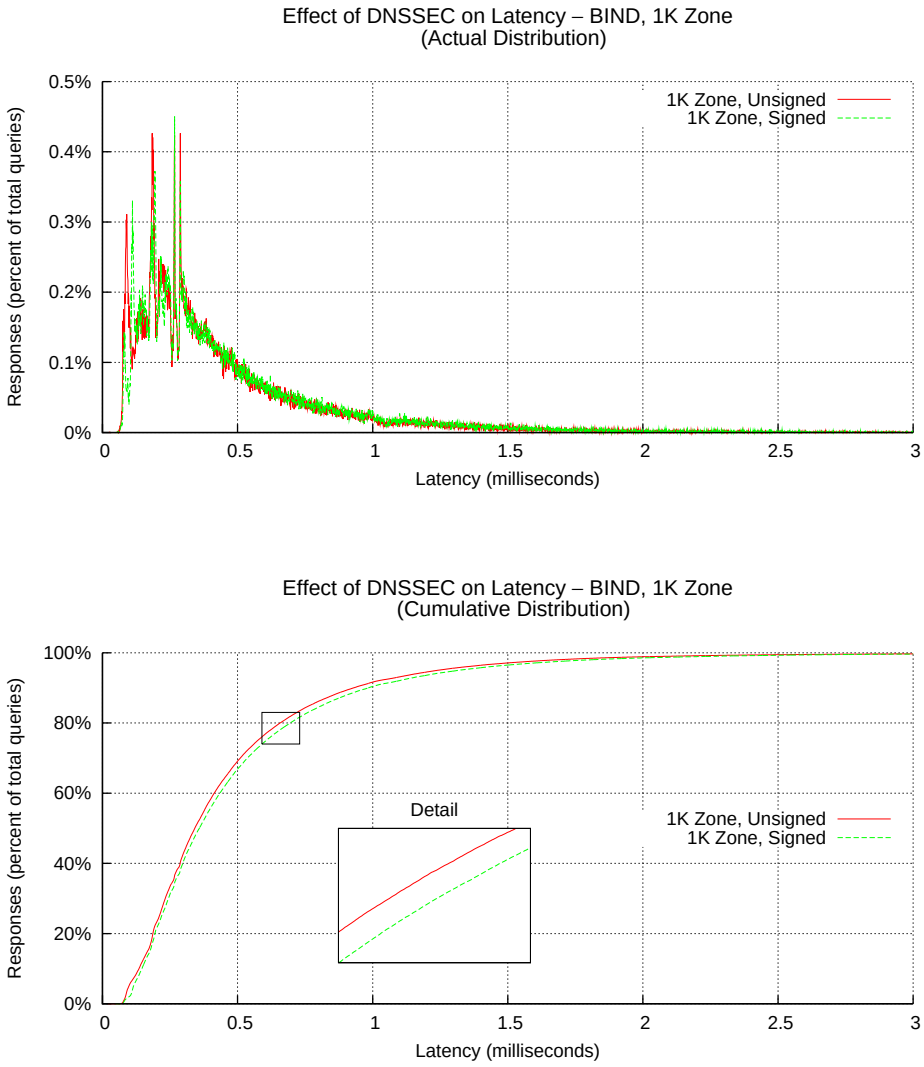

Figure 3.12: Effect of DNSSEC on latency – BIND, 1K zone.

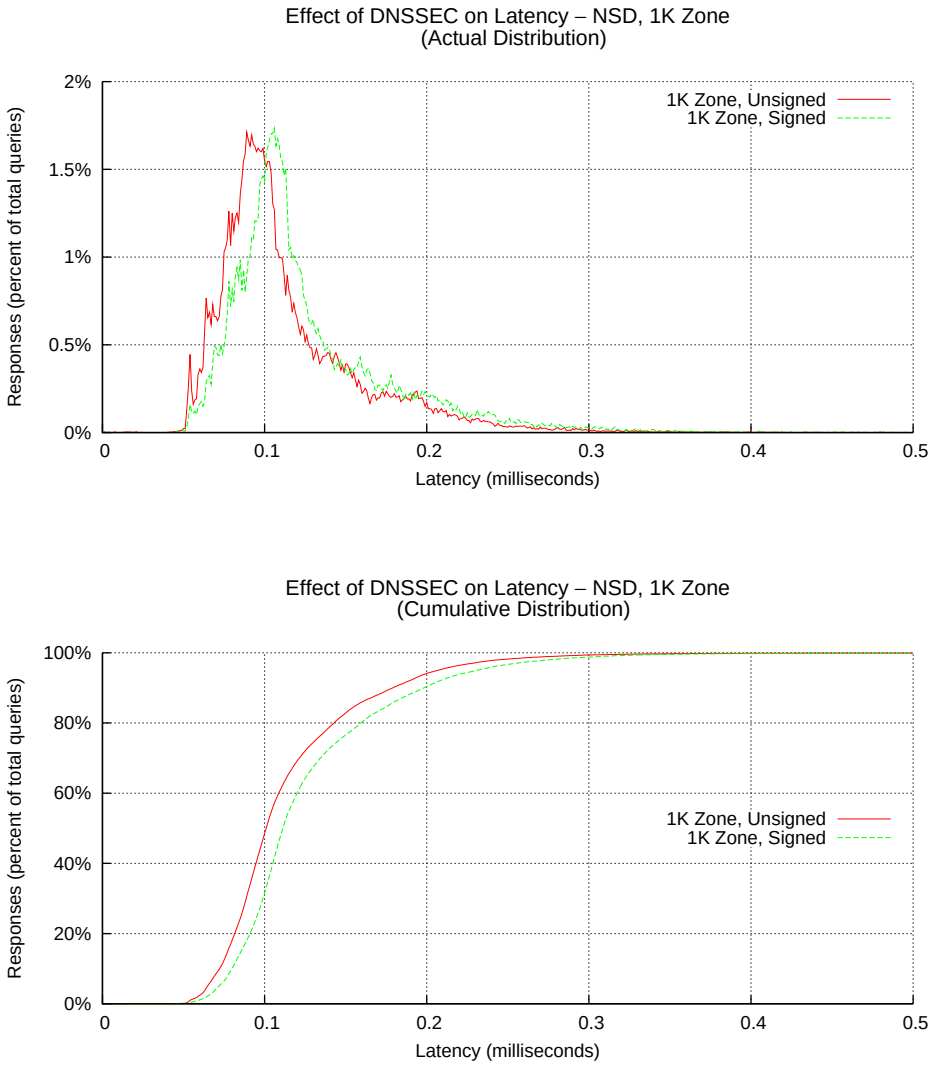

Figure 3.13: Effect of DNSSEC on latency – NSD, 1K zone.

In Figure 3.14 a tail of delayed responses starts to become apparent for BIND. The shift for NSD in Figure 3.15 also exhibits an attenuated tail, but it is tiny in comparison with BIND's. Note that these two plots – as well as the ones that follow in this section – have logarithmic x-axes that serve to render the distribution more clearly, but also have the effect of visually understating severe latency.

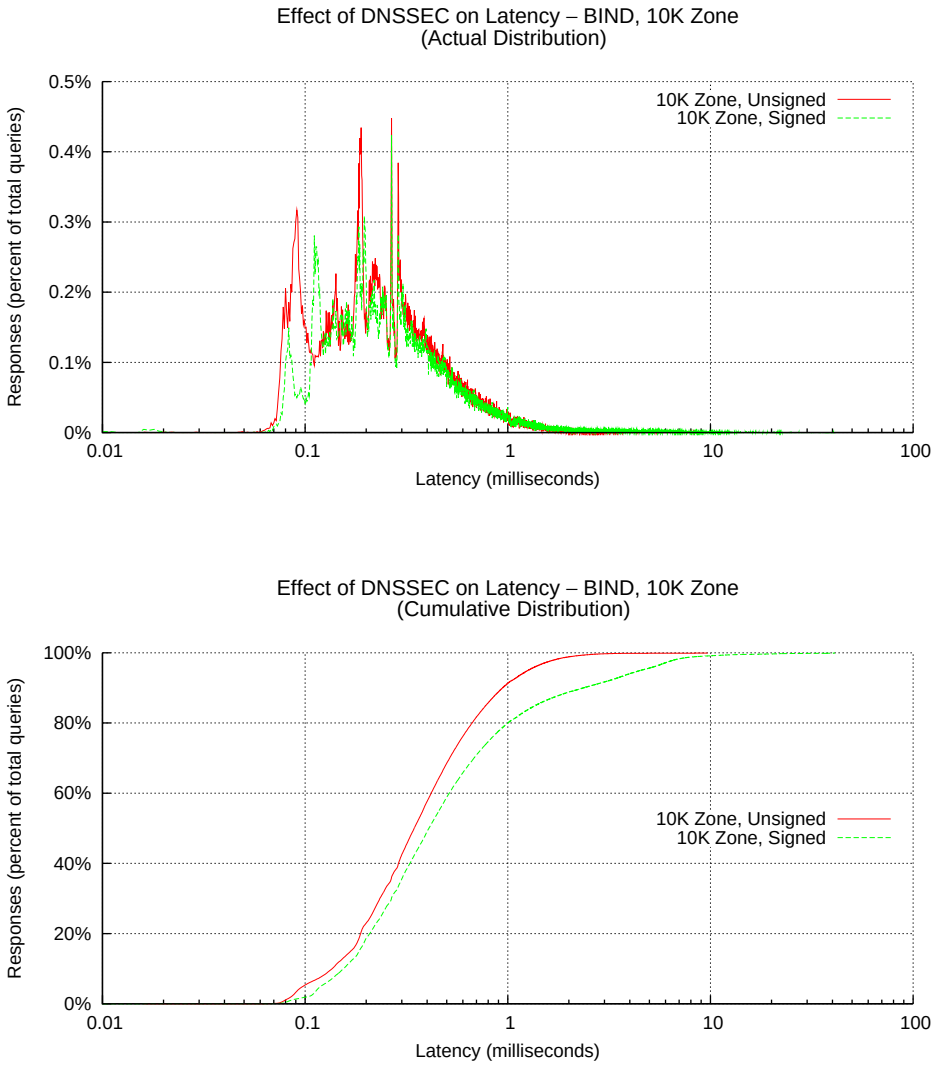

Figure 3.14: Effect of DNSSEC on latency – BIND, 10K zone.

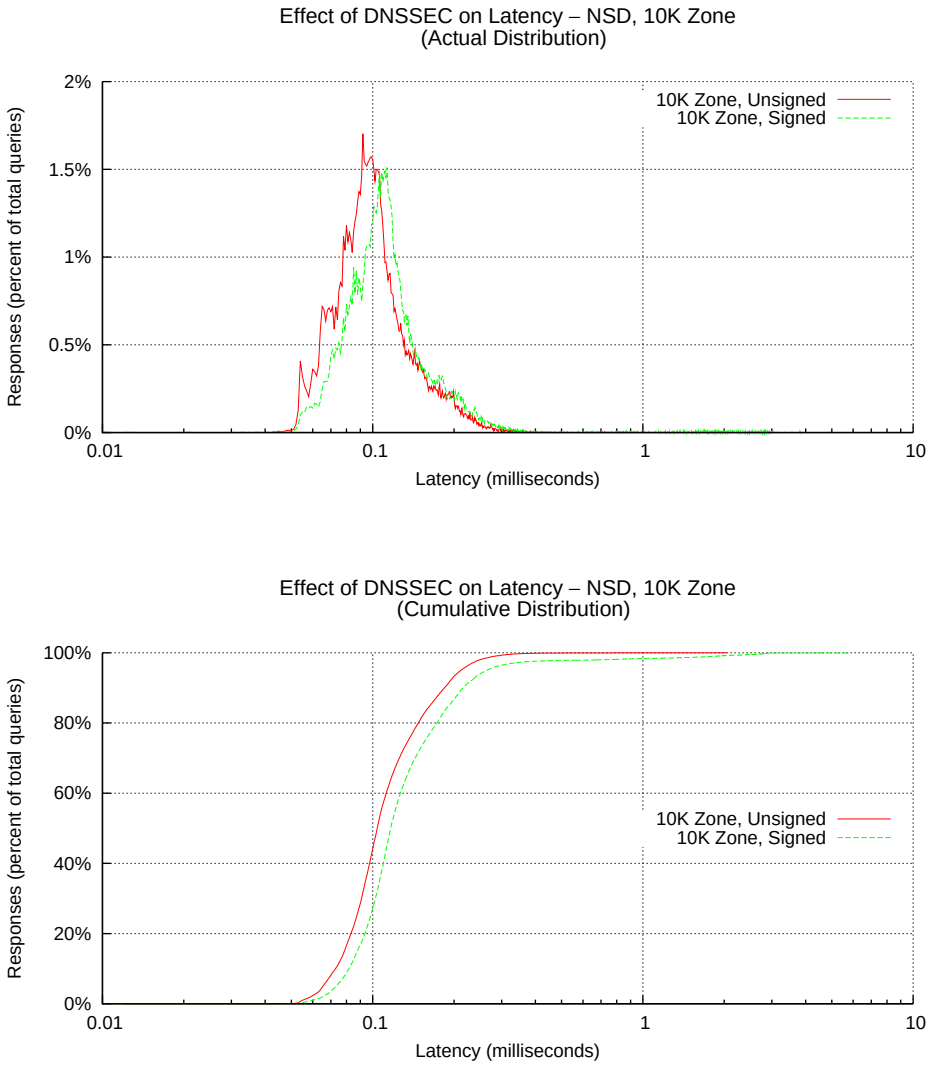

Figure 3.15: Effect of DNSSEC on latency – NSD, 10K zone.

Figures 3.16 and 3.17 show that latency starts to take a toll for the signed zone. Nearly 20% of queries to BIND go unanswered (within 4 seconds).

Figures 3.18 and 3.19 show that neither BIND nor NSD can sustain adequate performance when serving the signed 1M TLD root zone file. More than 80% of queries are dropped by BIND and nearly 22% are dropped by NSD.

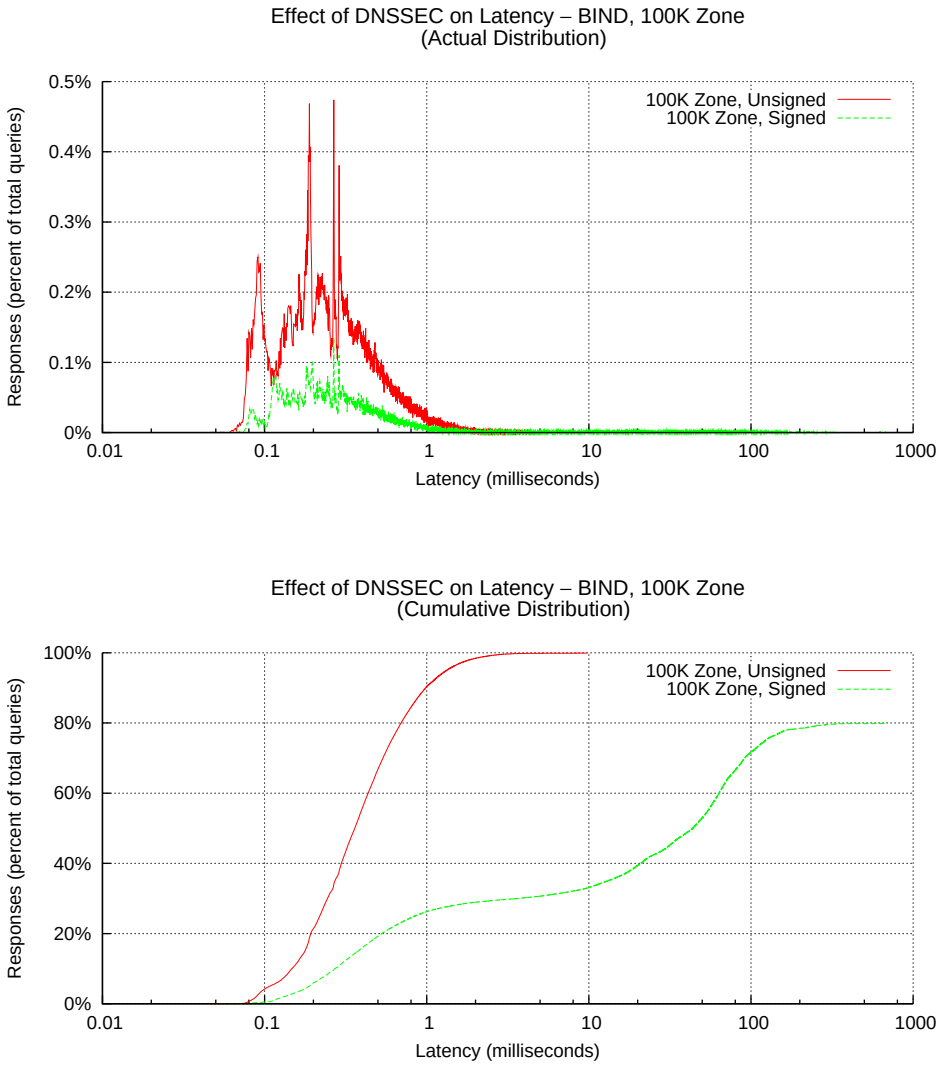

Figure 3.16: Effect of DNSSEC on latency – BIND, 100K zone.

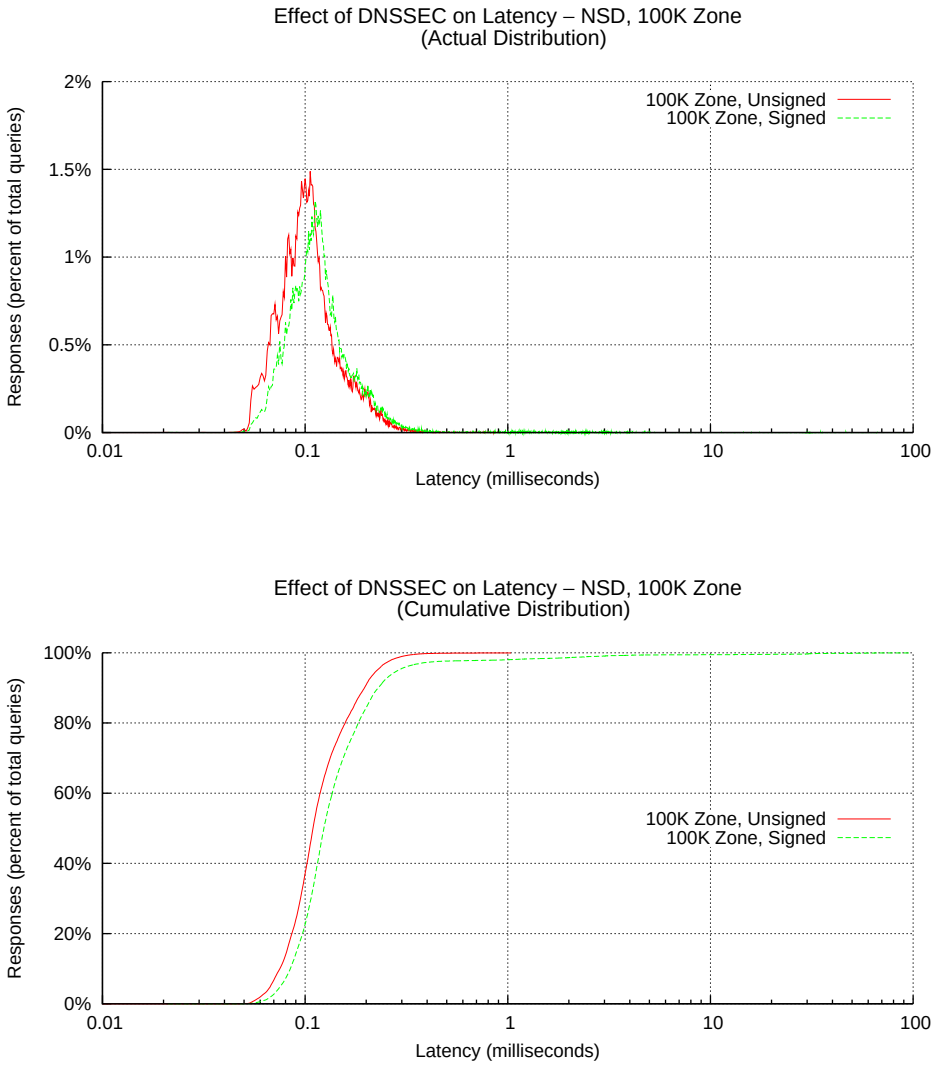

Figure 3.17: Effect of DNSSEC on latency – NSD, 100K zone.

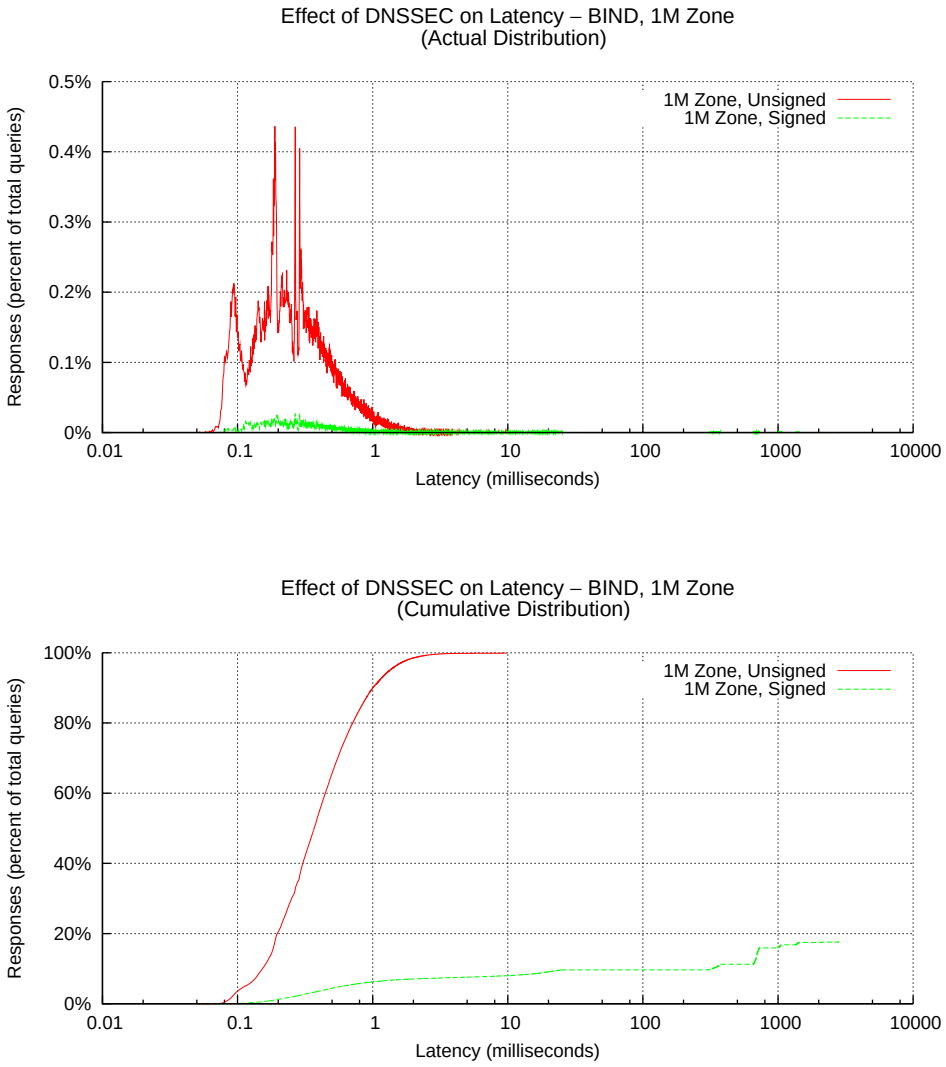

Figure 3.18: Effect of DNSSEC on latency – BIND, 1M zone.

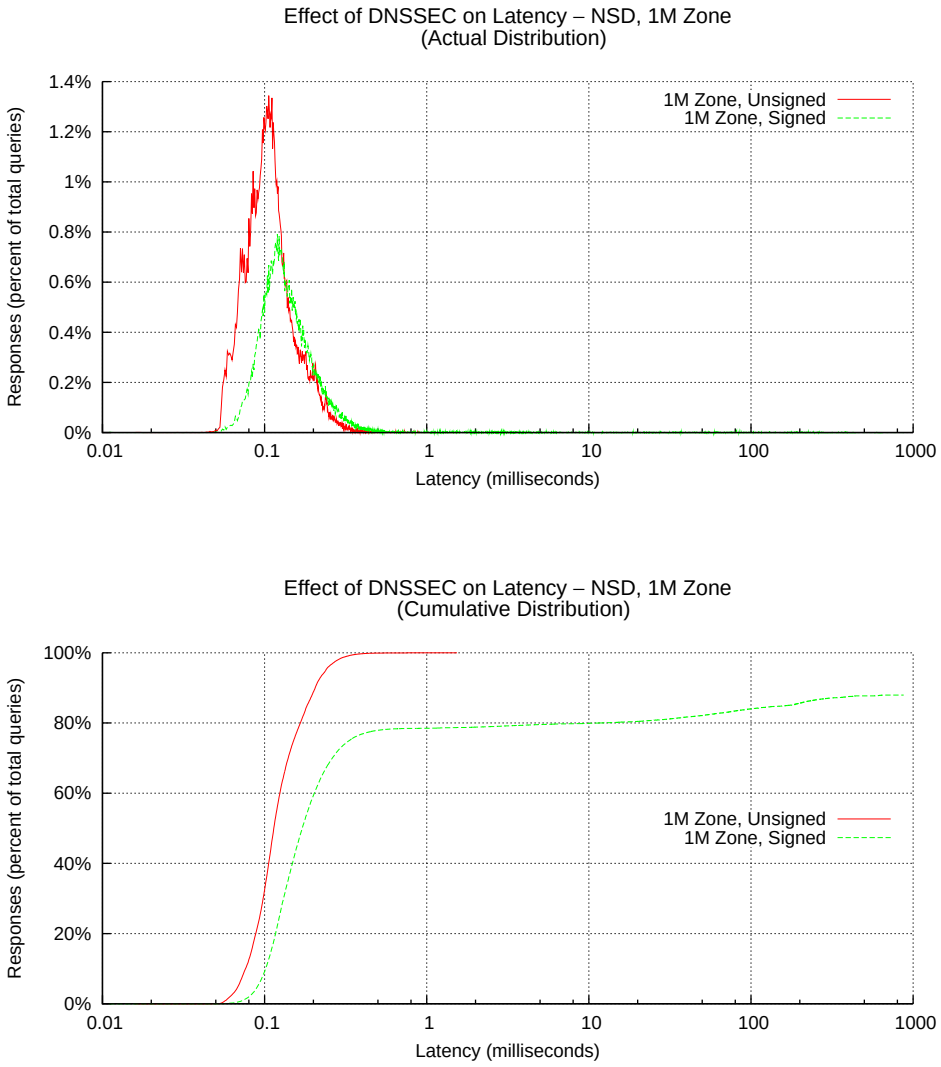

Figure 3.19: Effect of DNSSEC on latency – NSD, 1M zone.

To better understand how rapidly BIND's performance degrades with large signed zones, we examined BIND's performance at different query rates. Figures 3.20, 3.21, and 3.22 show the results. The clearest indication of how severely large signed zones penalize BIND's performance is evident in Figure 3.22: at just 500 queries per second, more than 40% of replies are delayed by over 100 milliseconds, and more than 3% percent are lost altogether.

ISC is aware of the results of these simulations, has reproduced the problem in their own lab, and is working to improve this shortcoming in BIND's performance. We know, for example, that they have already identified the code responsible for the slowdown when serving large signed zones.

We looked to see whether there were any obvious pattern to the late replies. One pattern we thought we might find was that the late replies were all DNSSEC replies, i.e., contained RRSIG RRs; however this was not the case. The ratio of DNS replies was consistent with the number of queries with the DO-bit set. Another possibility was that they might all have the same RCODE, but the RCODEs for the late replies were a typical cross-section.

It is clear from these examples that if the root zone is signed and becomes large, then a substantial number of additional servers will be required to distribute the load and to maintain adequate reserves to withstand traffic peaks and DDoS attacks. The precise number of servers required will need to be determined by capacity planning methods. With additional testing, it should be possible to arrive at an estimator in the form of:

> *For every x TLDs, y NSD hosts or z BIND hosts will be required to handle q queries per second.*

This would likely differ for signed and unsigned root zones.

3.2.4 Latency Differences Between Signed Zones with Sparse and Full DS Resource Records

Once the root is signed, another variable comes into play that could have a bearing on latency: the number of DS records that will be in the root zone. We tested zones with three different densities of DS RRs: 10%, 50% and 100%. It is considered best practice to have two DS RRs for each signed

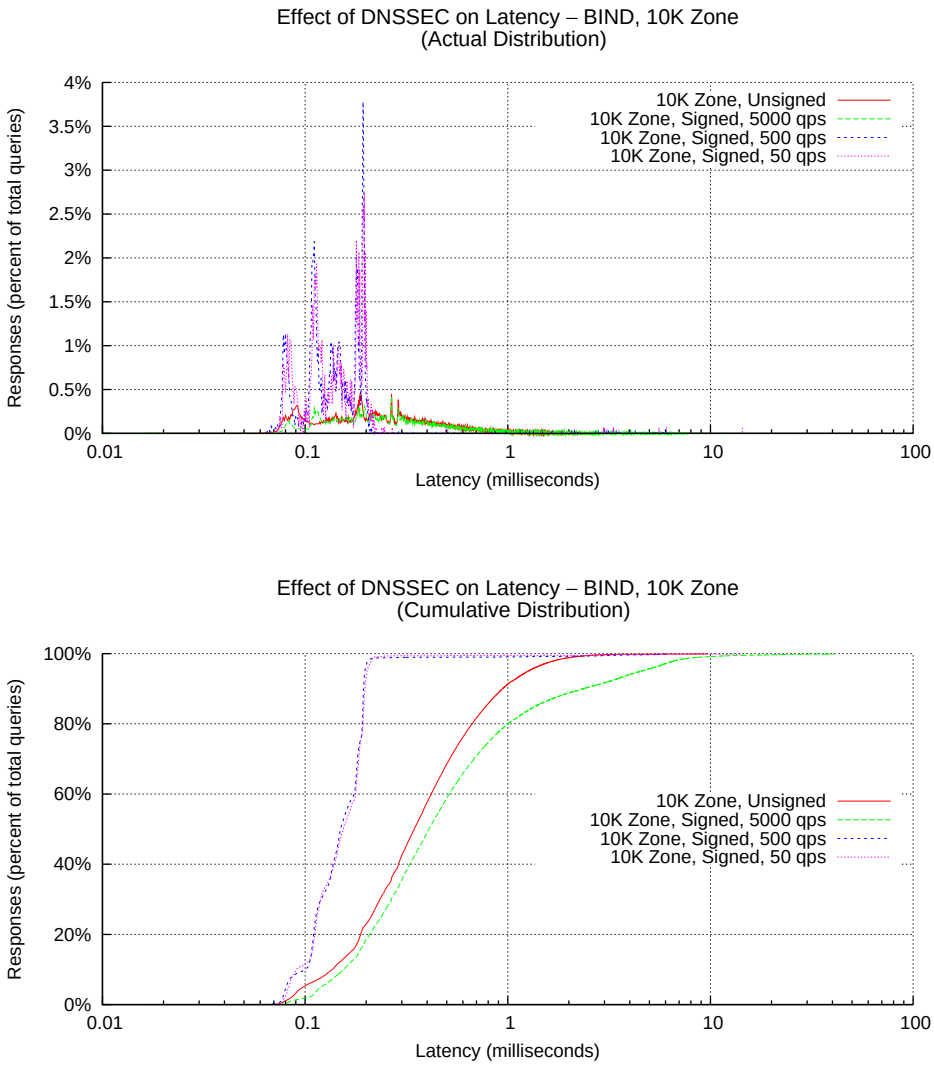

Figure 3.20: Effect of DNSSEC on latency – BIND, 10K zone, varying query rates.

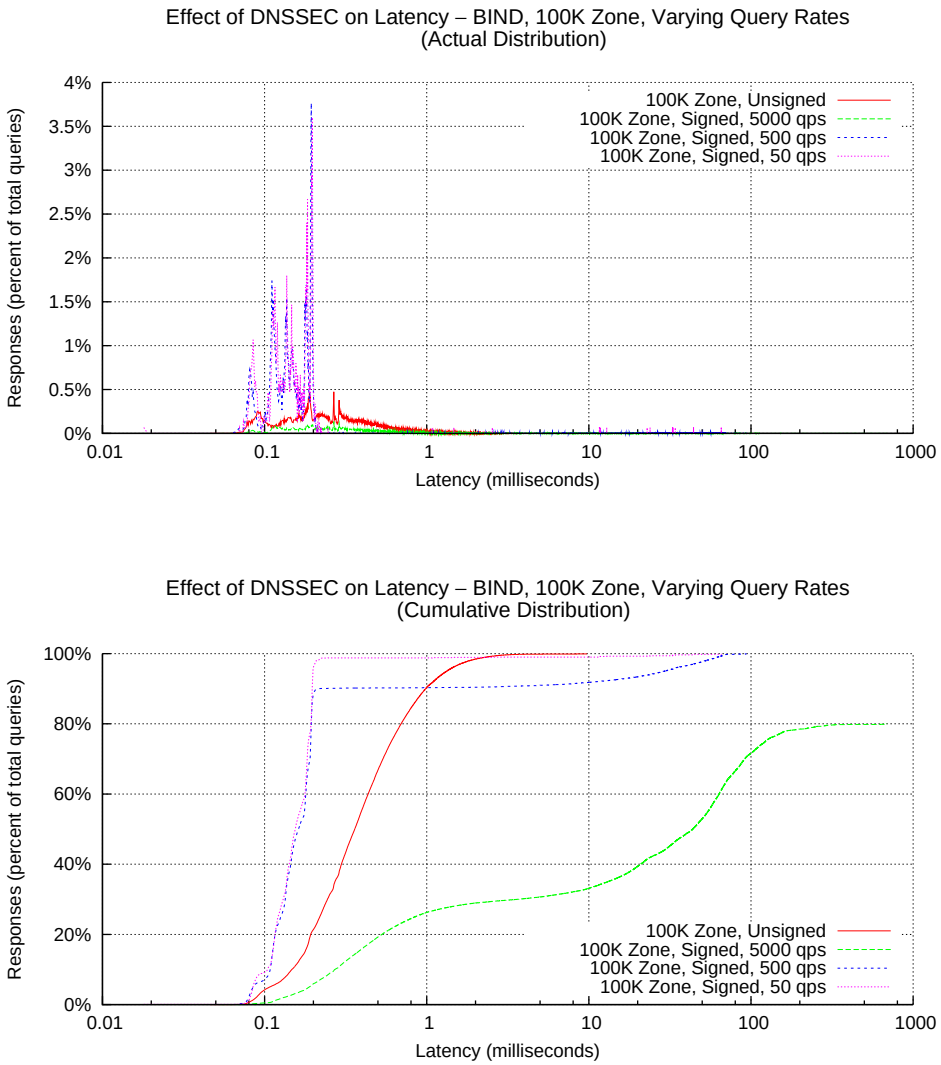

Figure 3.21: Effect of DNSSEC on latency – BIND, 100K zone, varying query rates.

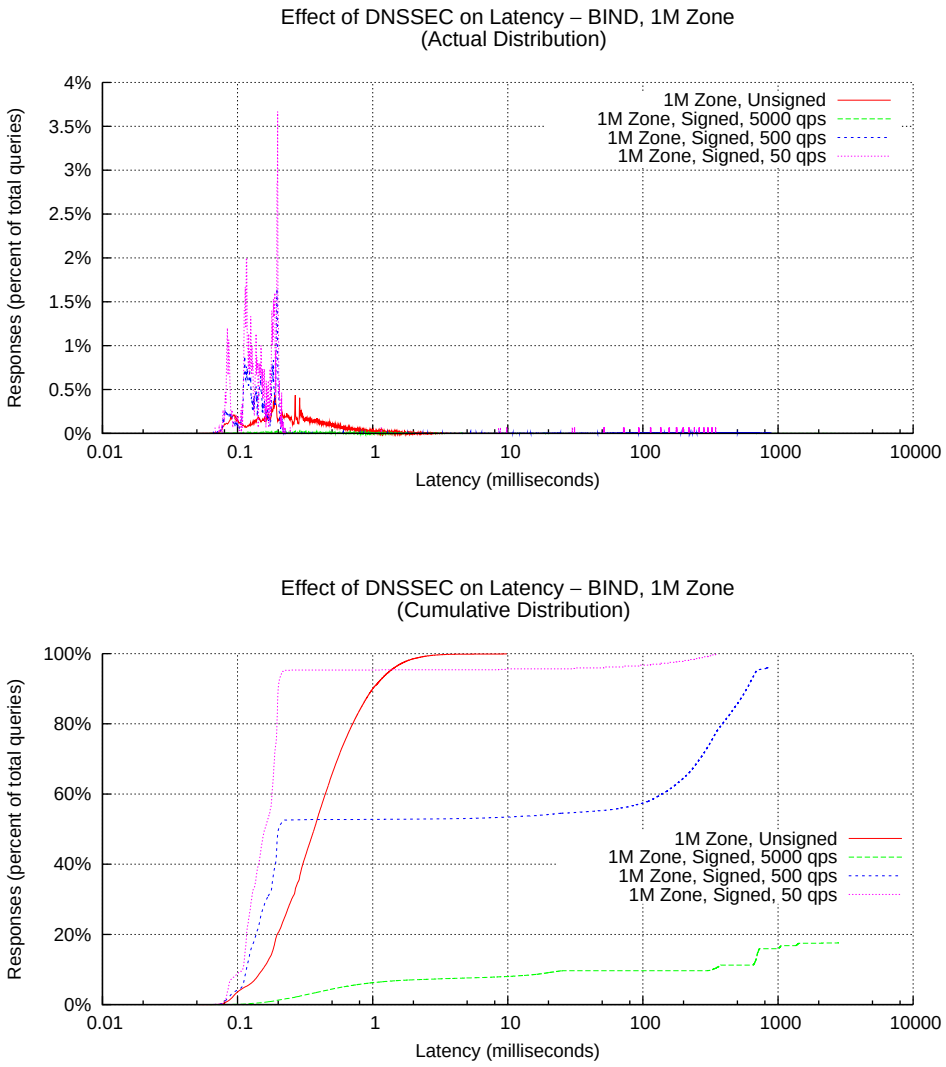

Figure 3.22: Effect of DNSSEC on latency – BIND, 1M zone, varying query rates.

delegation, so we used that as the basis for our testing. In other words, a zone with 50% DS RRs means half of the TLDs have two DS RRs each and the other half have none.

As with IPv6 glue, serving DS RRs involves some additional work by the server: referral replies become larger to accommodate the additional information, and some additional processing is needed.

We undertook to quantify the differences. Figures 3.23 through 3.30 show the results for signed zones of increasing size. In short, similar to IPv6 glue, the latency increases slightly as the number of TLDs with DS RRs increases, but the effect isn't very pronounced except for the largest zones.

There was one unusual result: the 1M zone with 10% DS RRs (S-6-DS10) caused *more* latency than the zones with 50% or 100% DS RRs with both NSD and BIND. This deviates from the pattern seen in the other zone sizes. We considered that this might be due to a testing error, but repeated trials appeared to rule this out.

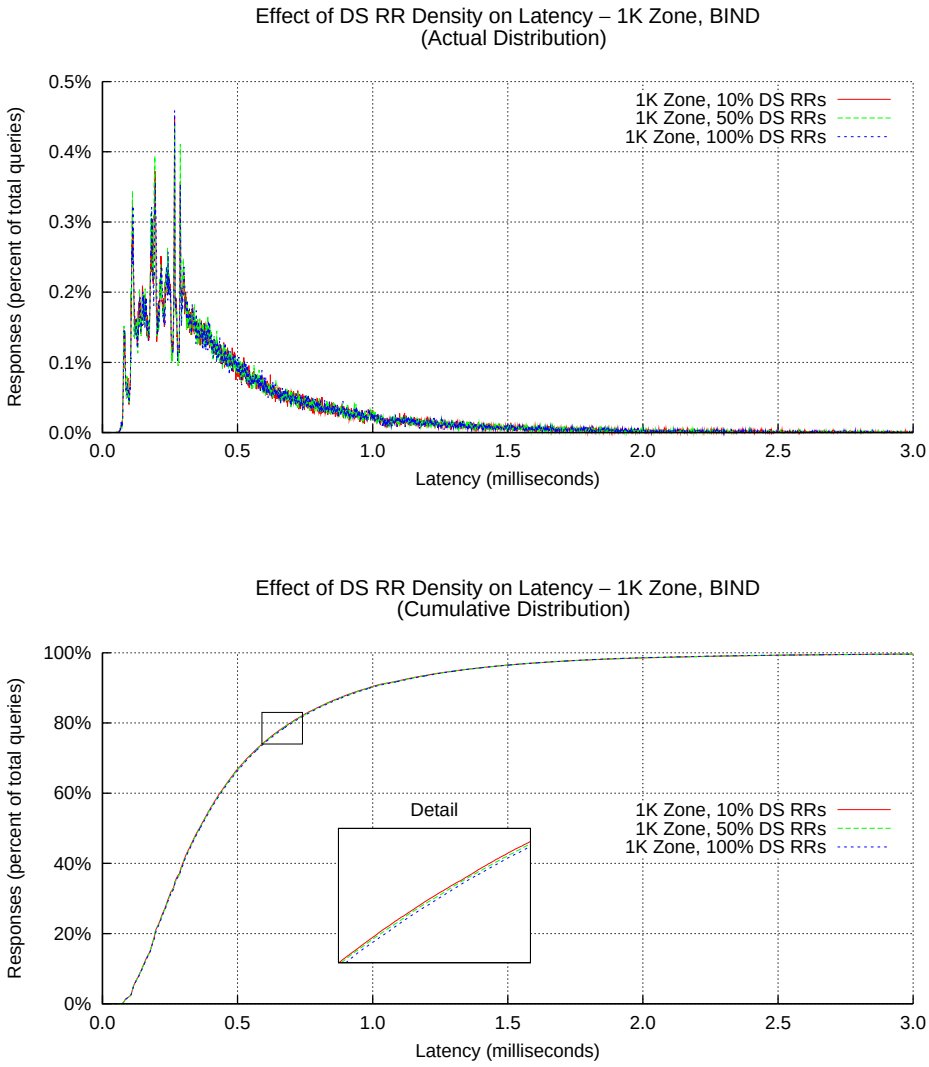

Figure 3.23: Effect of DS RR density on latency – 1K zone, BIND.

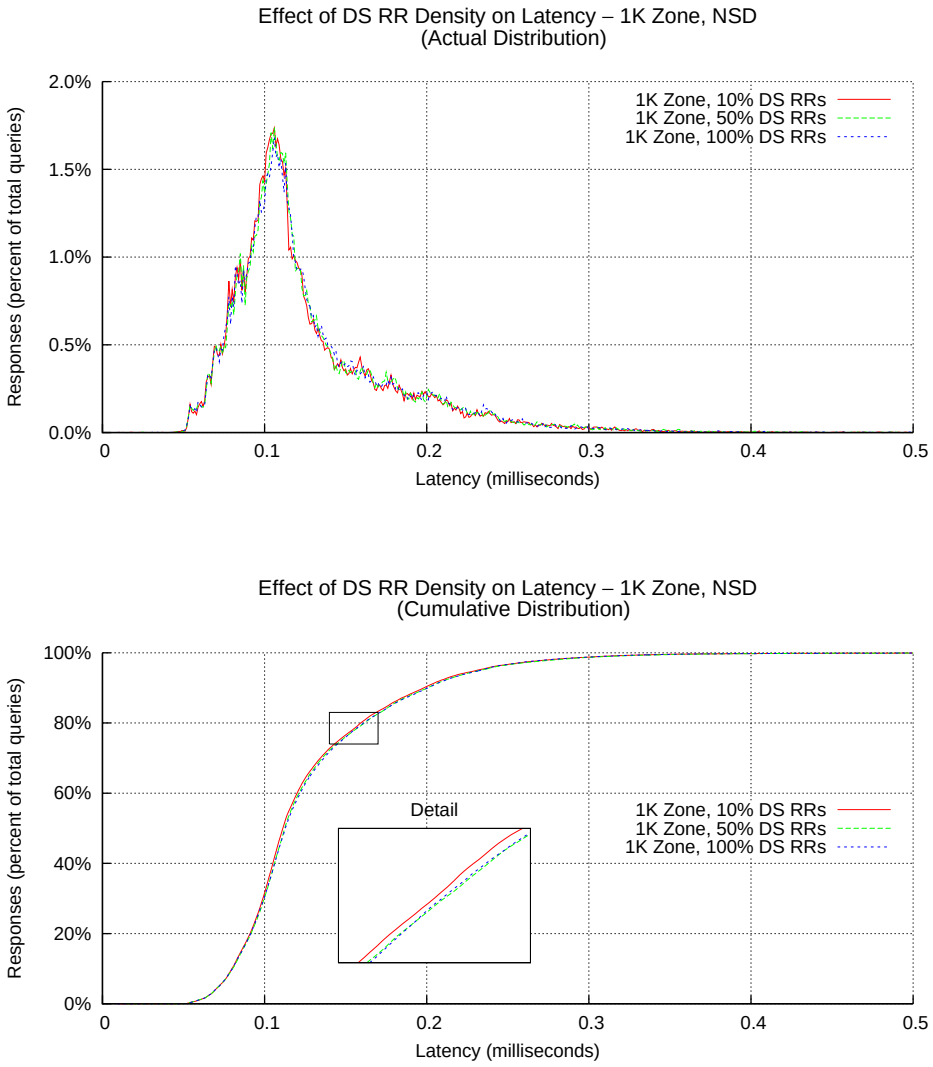

Figure 3.24: Effect of DS RR density on latency – 1K zone, NSD.

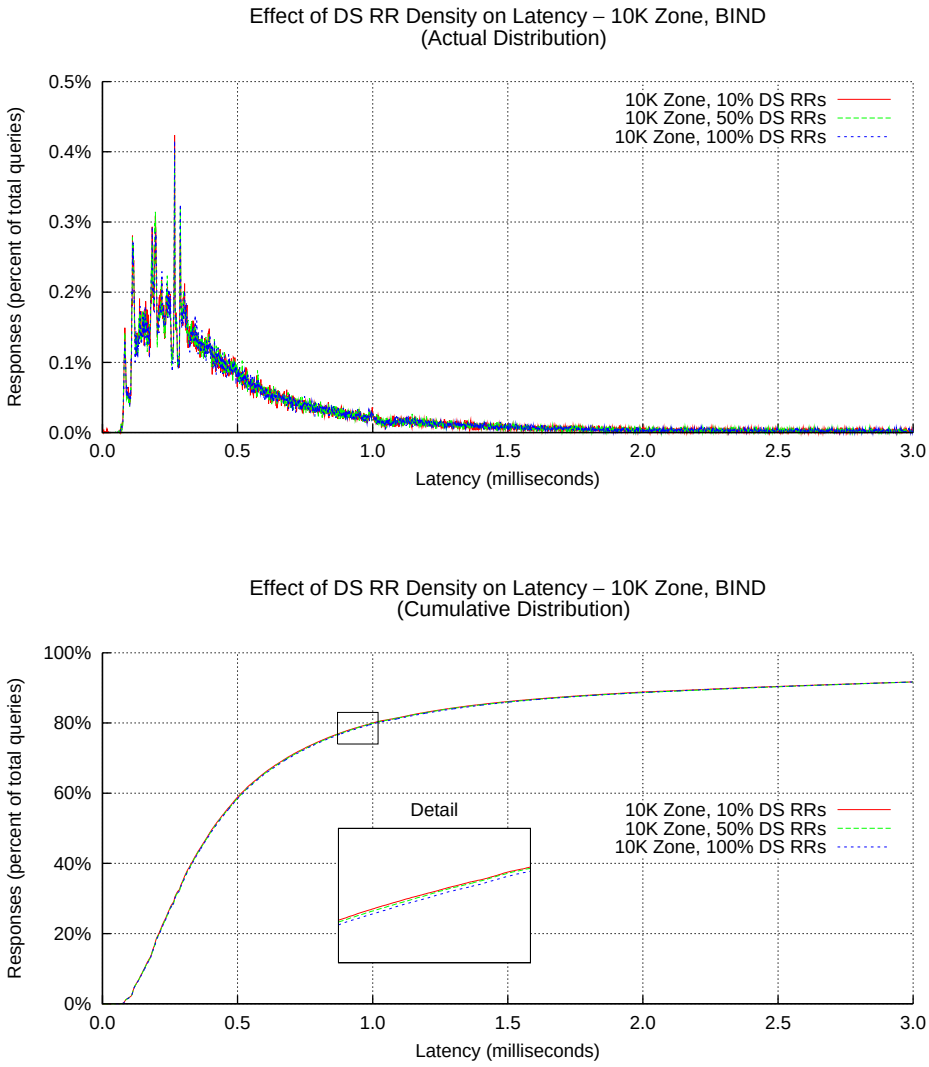

Figure 3.25: Effect of DS RR density on latency – 10K zone, BIND.

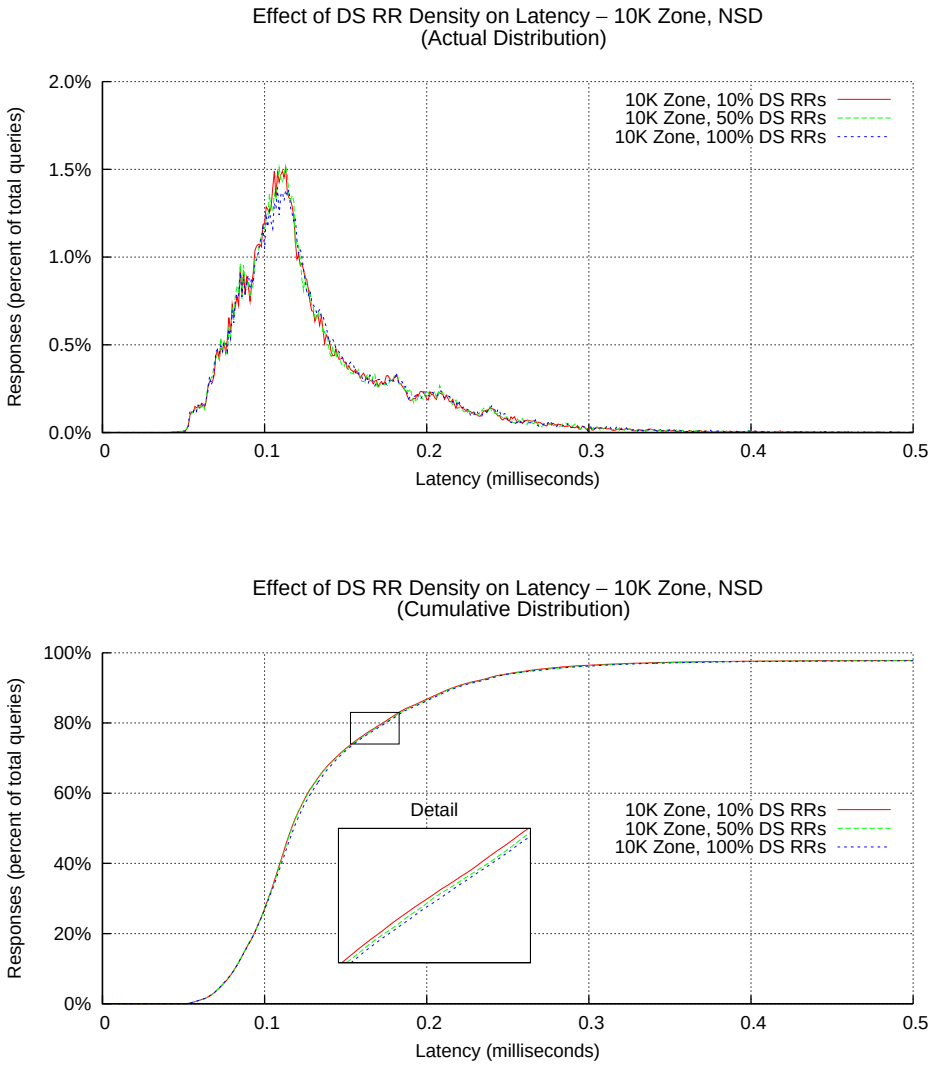

Figure 3.26: Effect of DS RR density on latency – 10K zone, NSD.

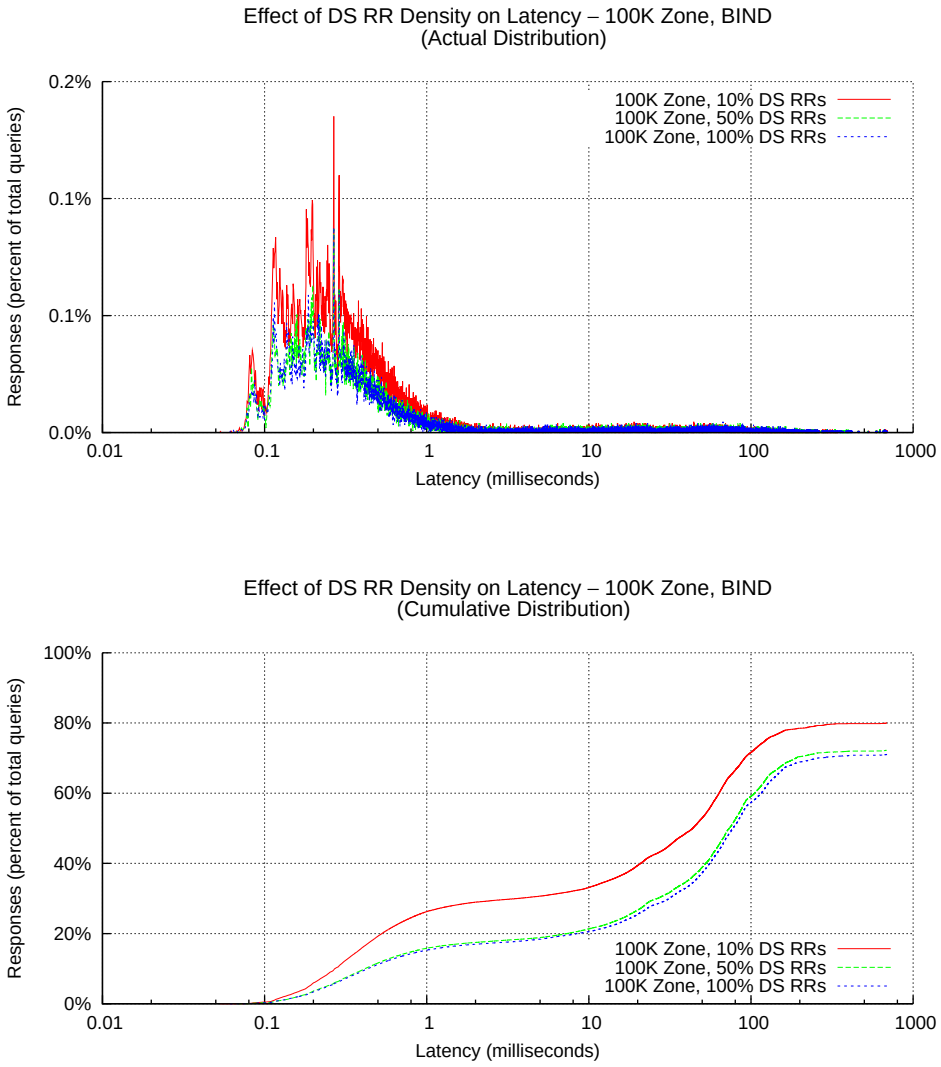

Figure 3.27: Effect of DS RR density on latency – 100K zone, BIND.

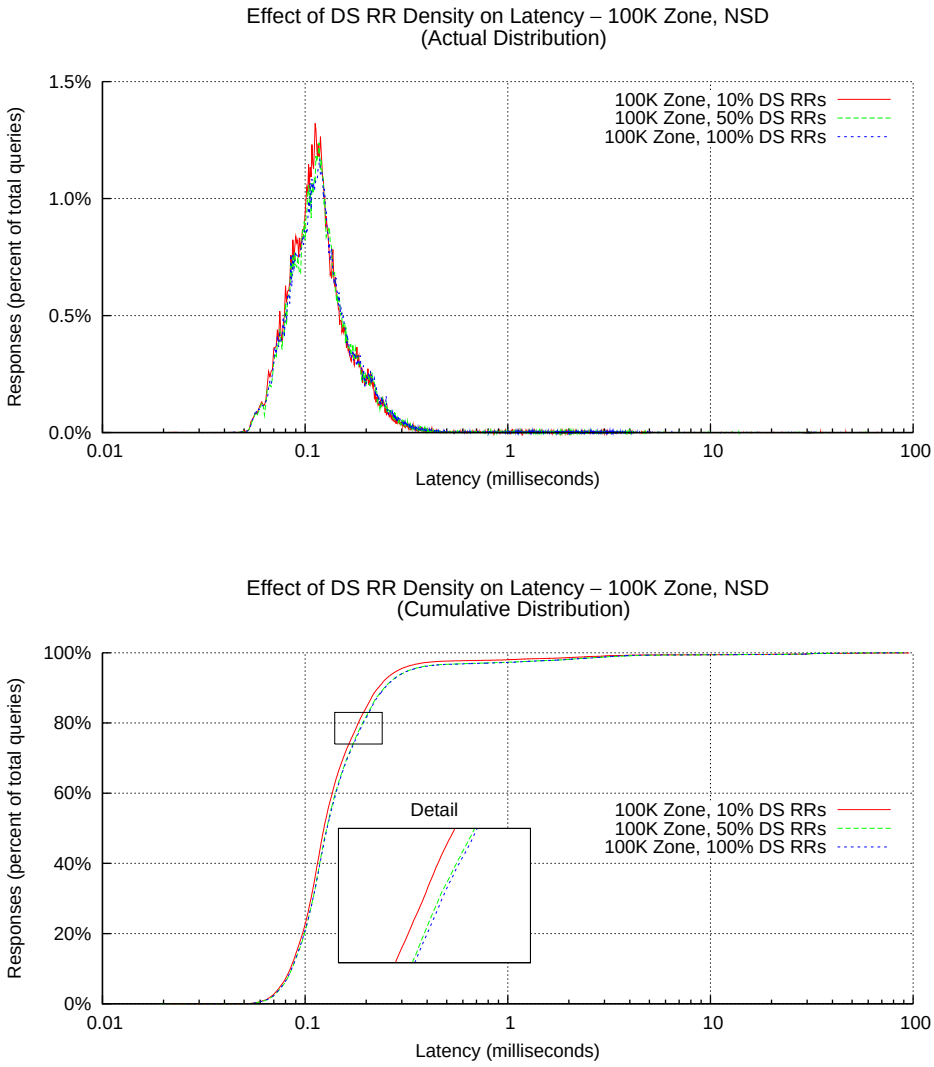

Figure 3.28: Effect of DS RR density on latency – 100K zone, NSD.

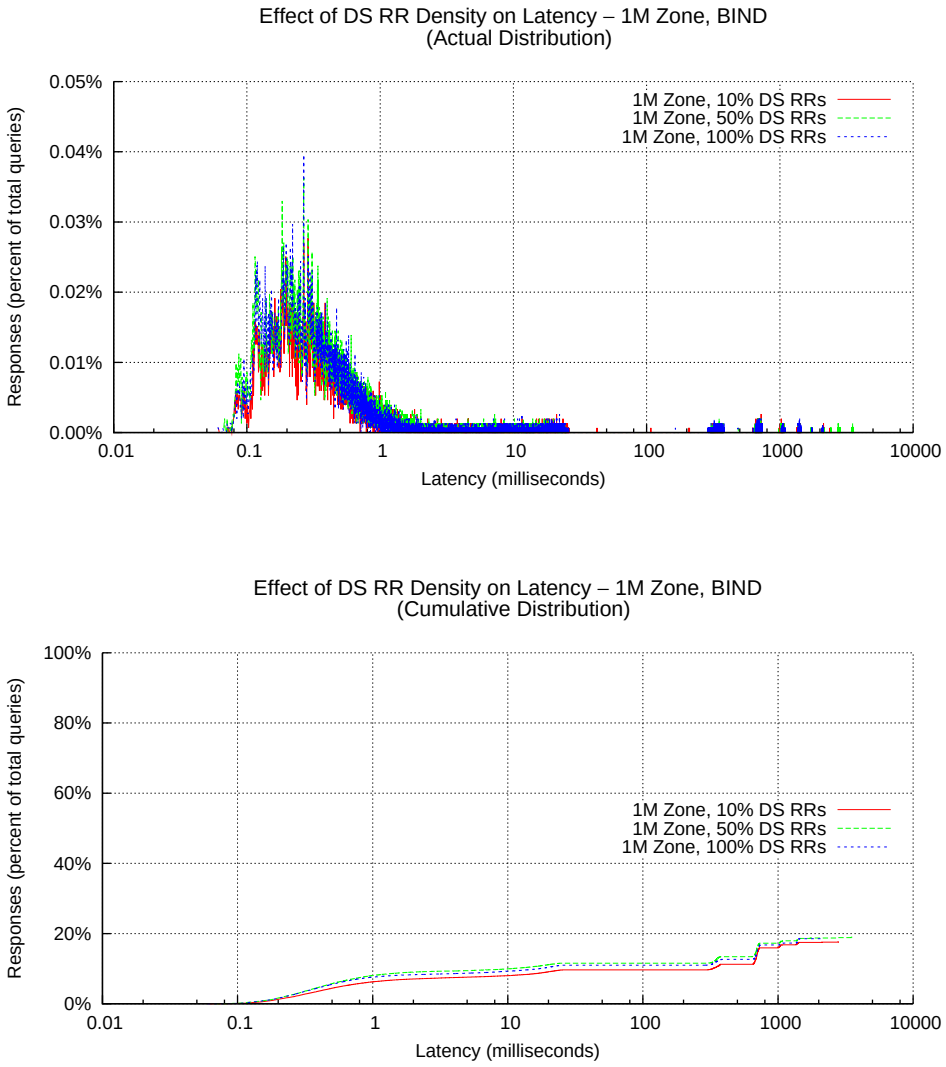

Figure 3.29: Effect of DS RR density on latency – 1M zone, BIND.

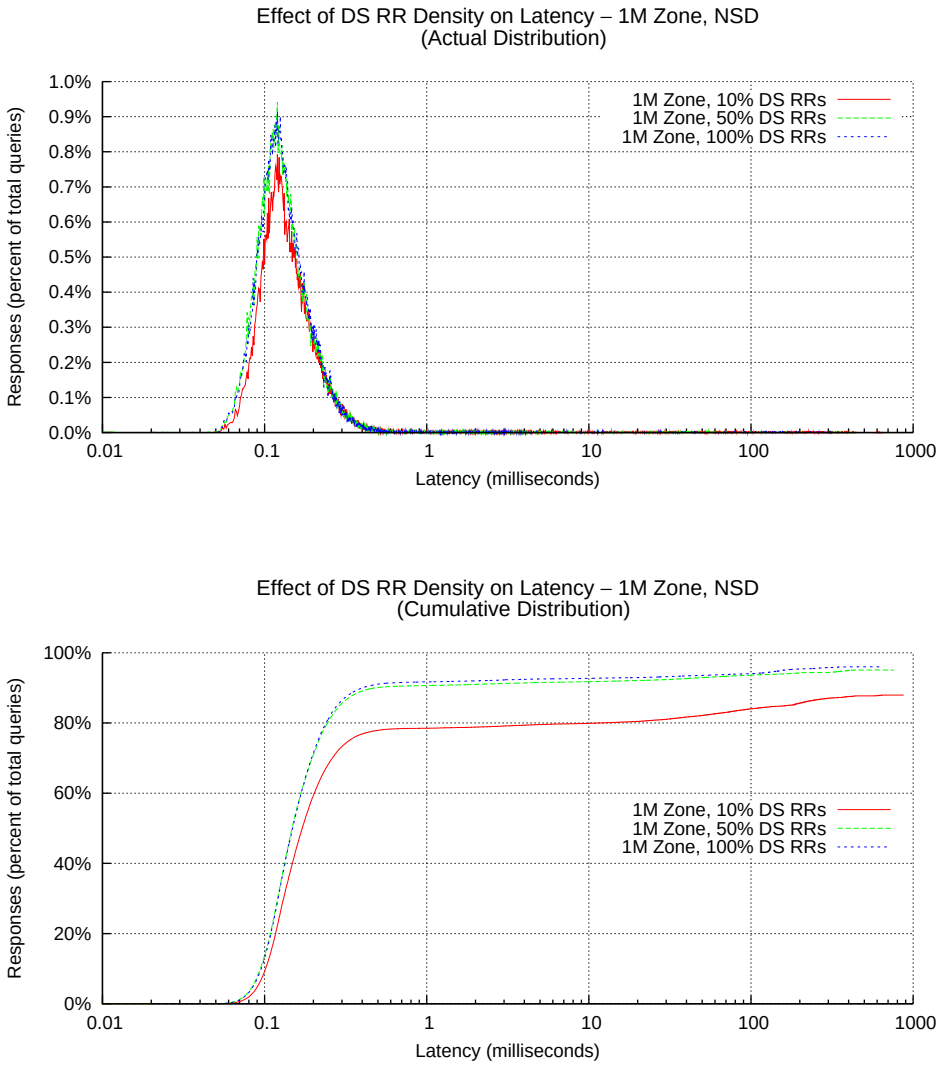

Figure 3.30: Effect of DS RR density on latency – 1M zone, NSD.

3.2.5 Latency Differences Between Queries Based On RCODE Distribution.

The root zone occupies a unique position in the DNS hierarchy. One of the consequences of this is that lookups for names that do not correspond with any TLD typically cause a query to be sent root servers. As a result, the root servers receive many millions of erroneous or misdirected queries per hour. Because the replies to these are typically either not cached, or are cached for only a short period of time, they are repeated so often that they constitute the bulk of queries to the root server.

Figures 3.31 and 3.32 show the differences in latencies that occur when some, many, and most of the queries are for names that don't exist in the root zone. The higher the ratio of queries for non-existent names, the lower latency of replies. Overall the differences are small between the latencies of replies to queries for predominantly non-existent names and queries for existing names.

3.2.6 Baseline Check

As a final exercise, we wanted to establish that there were no unexplained differences in latency between a sample of real queries and one generated by `querygen` to match the characteristics of the sample.

We randomly selected a 60-second period of queries to L-root captured during the DITL Data Collection Project, analyzed its characteristics, and then generated a new query stream with similar characteristics using `querygen`. We did not attempt to match all characteristics; for example, `querygen` always sets the RD-bit to zero and the query class (QCLASS) to "IN".

We rewrote the Ethernet MAC addresses and IP destination address of the original sample and replayed it using tcpreplay to instances of BIND and NSD serving a copy of the current root zone. We then replayed the generated version.

Figures 3.33 and 3.34 show the differences in latencies between the original and generated query streams for BIND and NSD respectively. The first plot of each figure shows the actual distribution, i.e., a histogram of the observed latencies. The second plot shows the cumulative distribution. As can be

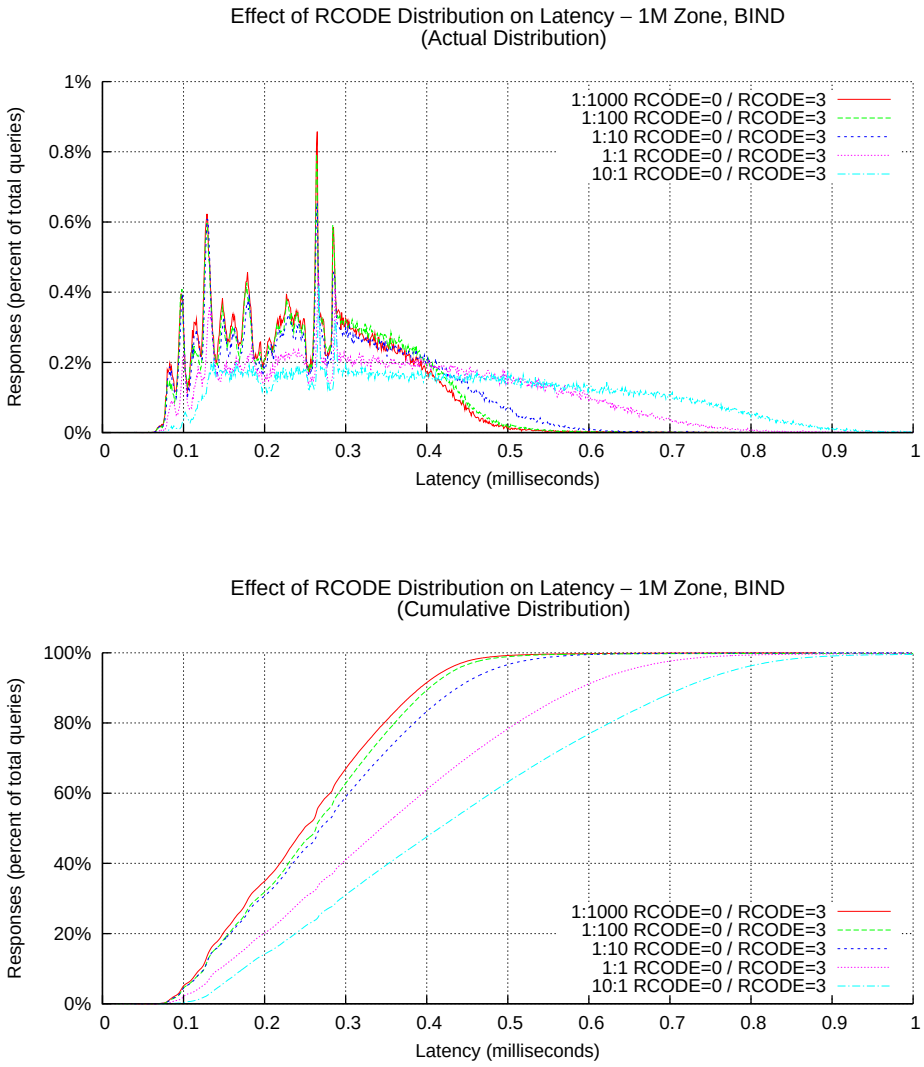

Figure 3.31: Effect of RCODE distribution on latency – 1M zone, BIND.

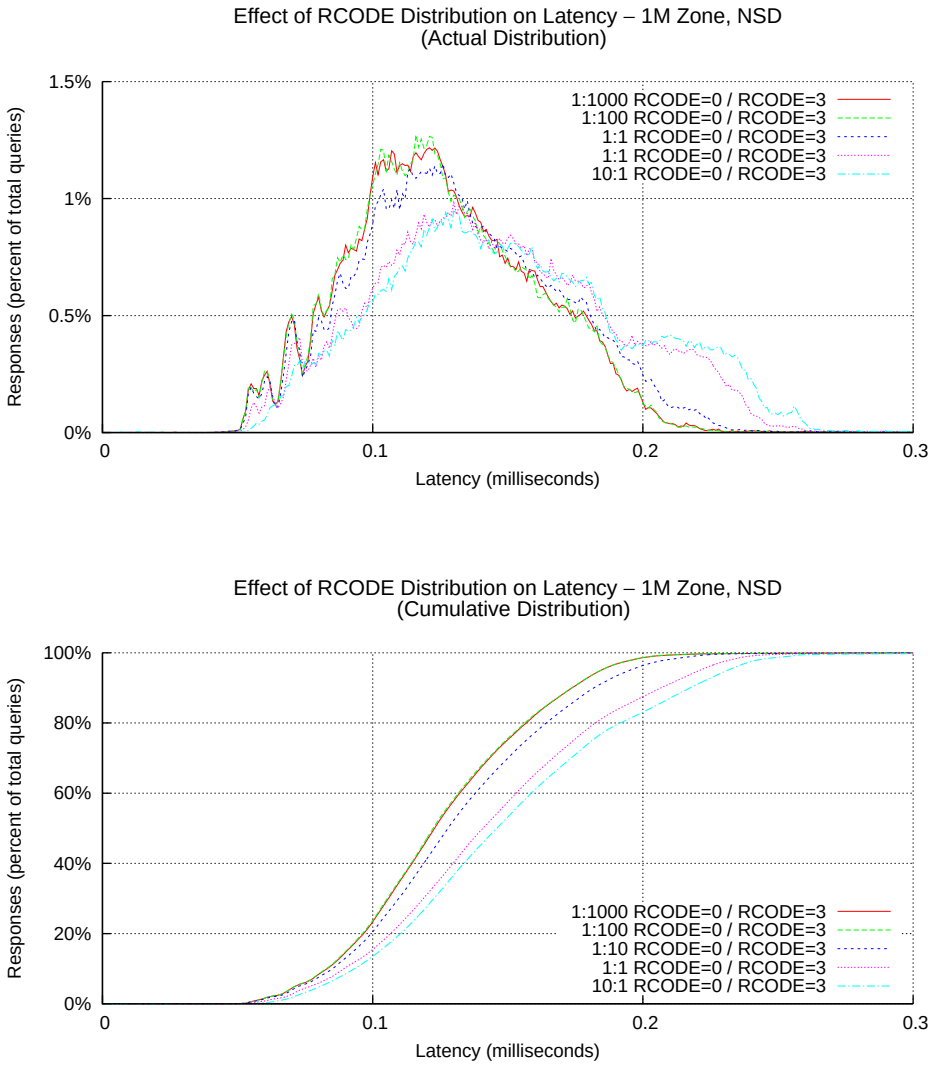

Figure 3.32: Effect of RCODE distribution on latency – 1M zone, NSD.

seen, any differences are small. Both query streams have an average query rate of around 5000 queries per second.

We repeated the comparison with BIND and NSD serving large root zone files, in this case copies of the signed one million TLD root zone. Figures 3.35 and 3.36 show the differences are more noticeable, but that they are still comparatively small.

These results do not demonstrate conclusively that there are no functionally significant differences between the original query stream and the artificially generated copy; they serve as *prima facie* evidence that any differences are likely to insignificant.

One final baseline check we thought made sense was to see whether there were significant variations between successive replays. The results in Figures 3.37 suggest there is not.

3.3 Summary

These tests show that all augmentations being considered for the root zone will have *some* impact on latency. In general, the latency is orders of magnitude smaller than the latency of the links between the root servers and centrally-positioned resolvers. Even if IPv6 is universally deployed and the root zone swells to a million TLDs, the additional time imposed by increased server latency will be small.

The one area of concern is the combination of DNSSEC and a large root zone. A signed root with a million TLDs will likely require many times the number of servers deployed today to maintain the same operational capacity and headroom, even if NSD is used exclusively.

The performance changes we've seen during this analysis raises the possibility that name servers with large DNSSEC-signed zones might be particularly vulnerable to directed attacks. We've seen that for BIND, signing a one million TLD root zone causes BIND's performance capacity to drop by over 90%, even though response sizes increase by less than double. The picture is better with NSD, but it is still evident that the performance suffers disproportionately.

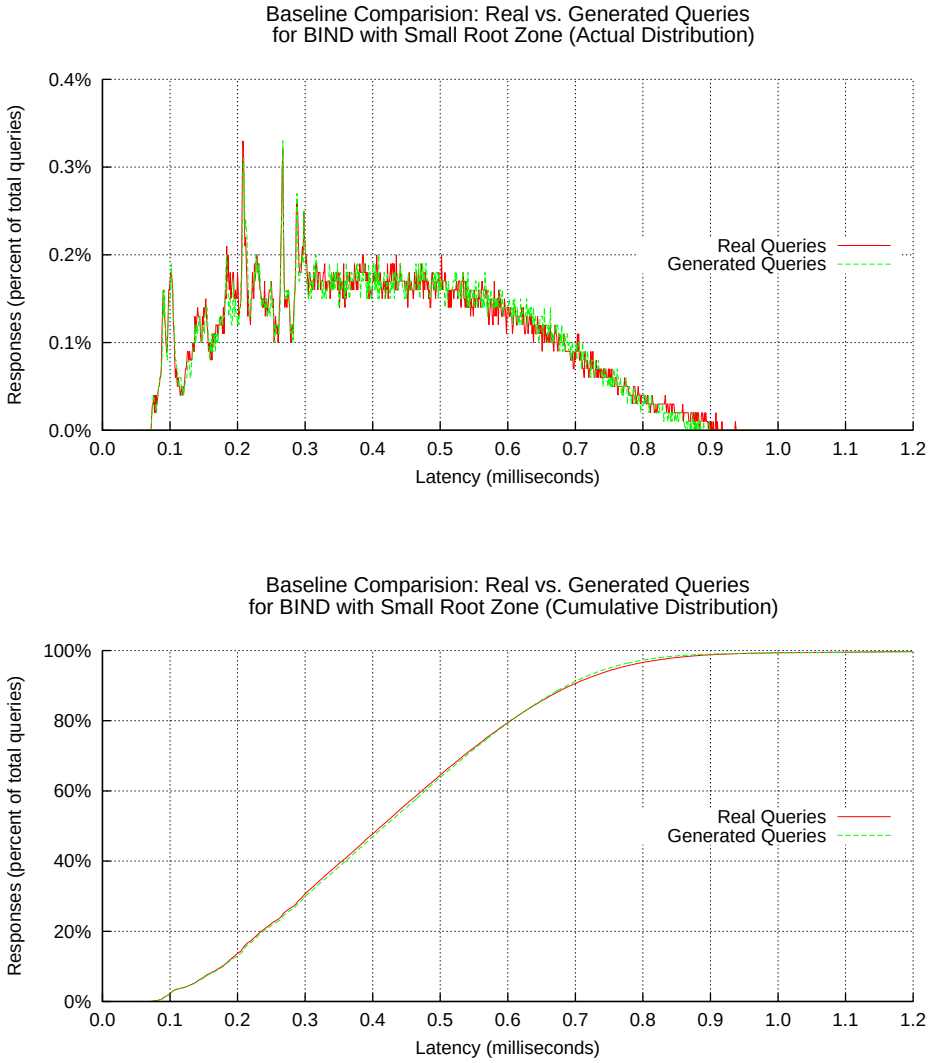

Figure 3.33: Baseline comparison between real and generated query streams – small root zone, BIND.

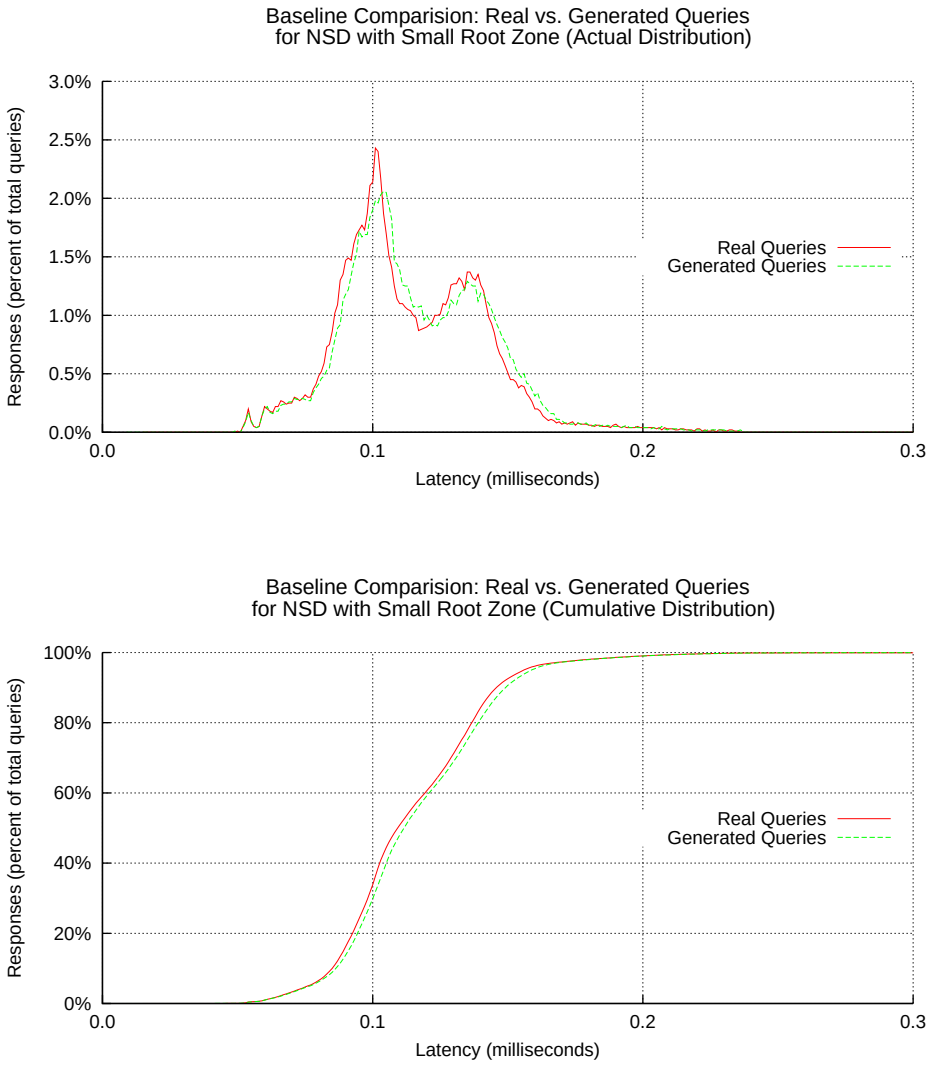

Figure 3.34: Baseline comparison between real and generated query streams – small root zone, NSD.

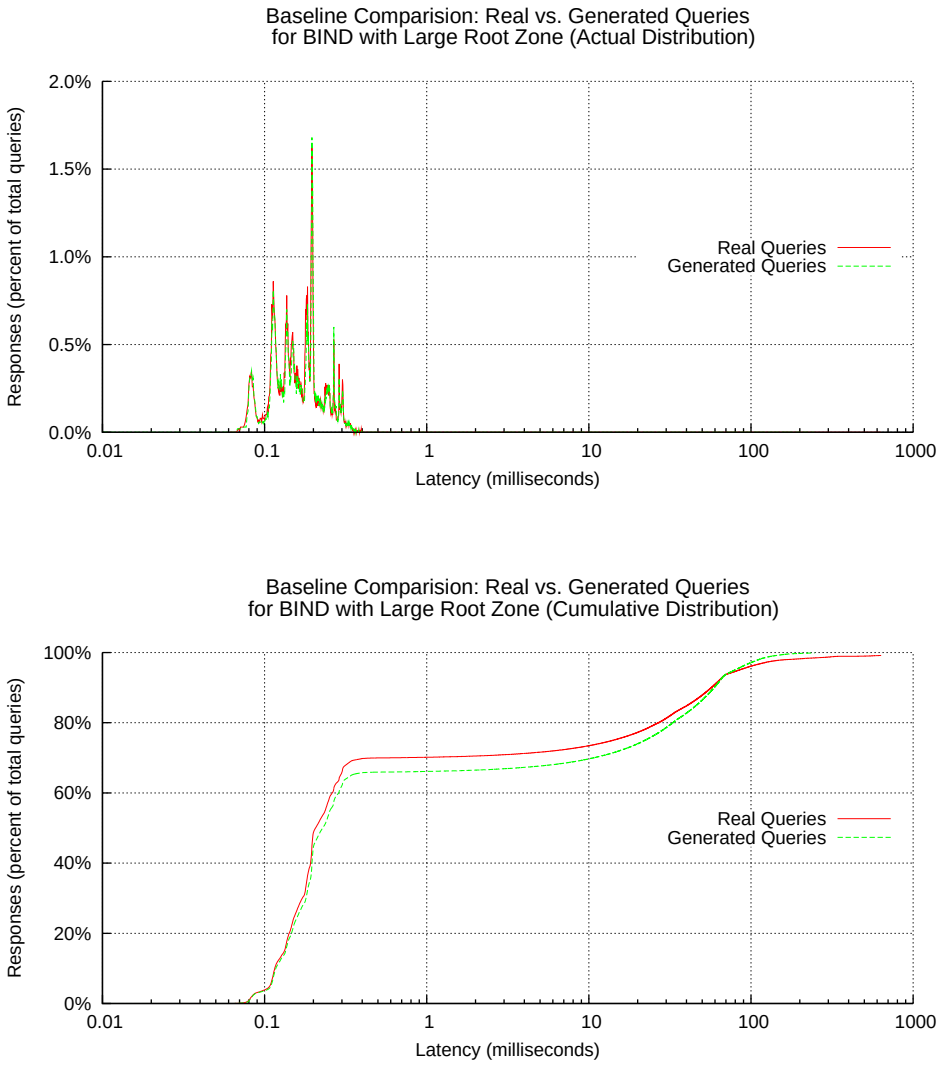

Figure 3.35: Baseline comparison between real and generated query streams – large root zone, BIND.

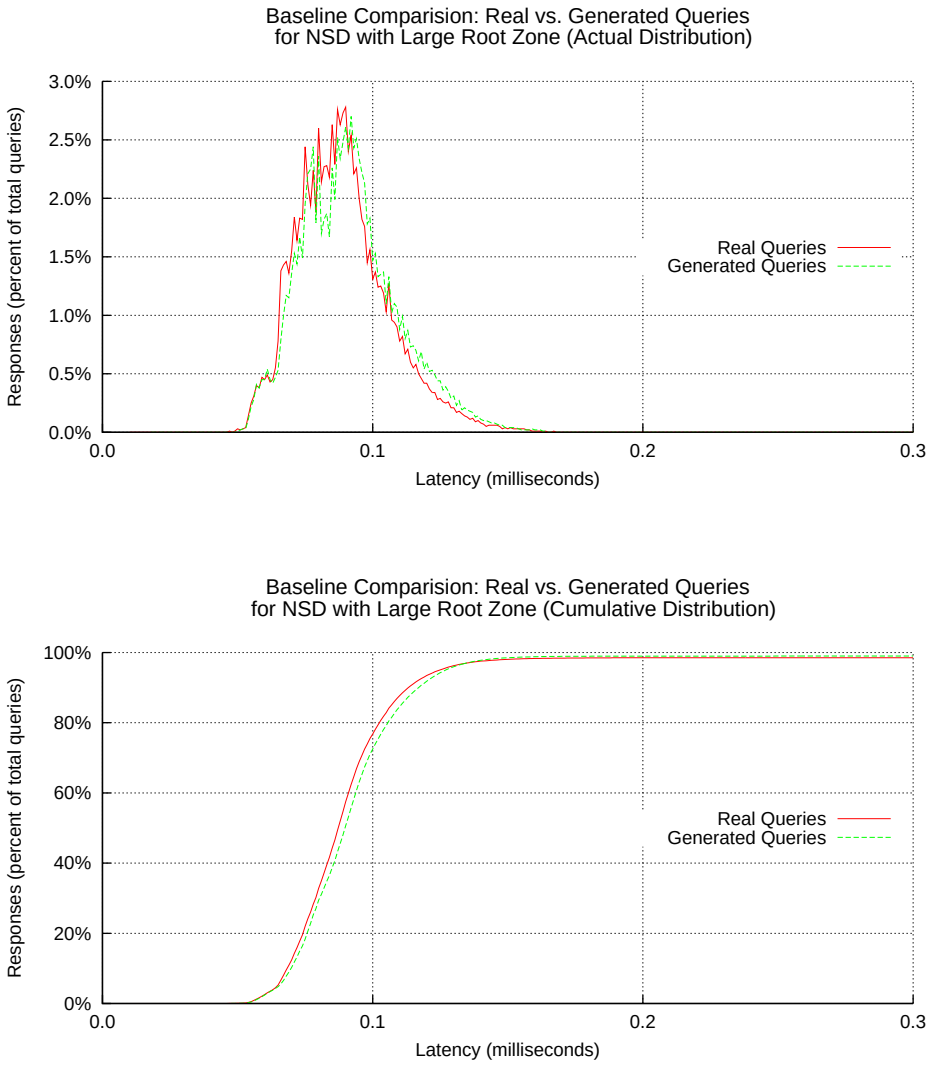

Figure 3.36: Baseline comparison between real and generated query streams – large root zone, NSD.

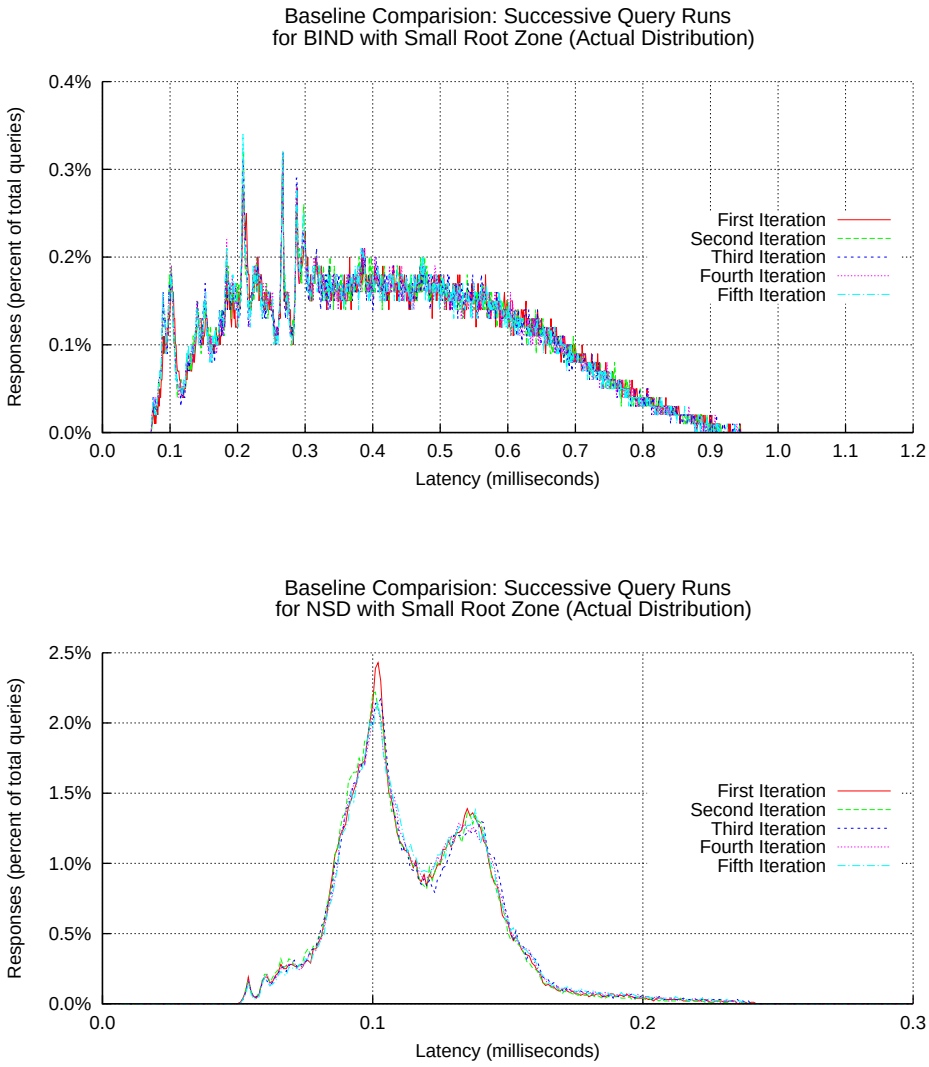

Figure 3.37: Differences between successive trials – BIND and NSD.

Chapter 4

Impact on Zone Reload and Server Restart Times

This task examines the impact of an increased number of TLDs in the root zone on the amount of time required to load and reload it. We were asked to consider the following:

> *Determine by way of experimentation how root zone reload and name server restart times are affected by increased root zone size. Key variables to examine are:*
>
> a. *Addition of IPv6 addresses for all name servers in the root zone*
> b. *Whether the zone is DNSSEC-signed (using the anticipated DNSSEC keys and key sizes)*
> c. *Percentage of TLDs have DS RRs with at least 10%, 50%, and 100% considered.*

4.1 Setup

For this task we use the same hardware, software, and zone files as in the first task (Chapter 2). As before, we use the same test harness to:

	Zone Size (#TLDs)				
Zone Type	1K	10K	100K	1M	10M
U-4-DS0	<1	<1	8	87	950
U-6-DS0	<1	<1	11	113	1153
S-6-DS10	<1	<1	14	157	1581
S-6-DS50	<1	<1	16	170	1723
S-6-DS100	<1	2	17	190	1911

Table 4.1: load time (in seconds) vs. zone size – BIND.

1. start the name server,
2. measure the amount of time until the name server returns a correct response for a resource record at the end of the zone file,
3. and record the results.

To measure the reload time, our harness then loads a zone file that differs only in the RDATA field of a resource record at the end of the zone. The reload is considered complete only when the server returns a correct answer for the changed resource record.

In this task, times are measured with a minimum resolution of one second. Where observations fall below the minimum resolution, the results are considered inconsequential. From an operational perspective they are practically instantaneous.

4.2 BIND Results

The raw data for load times are shown in Table 4.1. Note that for all but the largest of the 1K and 10K TLD cases, BIND begins serving from the end of the zone within one second or less.

Load and reload times are generally linearly proportional to the number of resource records in the zone.

Figure 4.1 plots the time required to load successively larger zones. Since

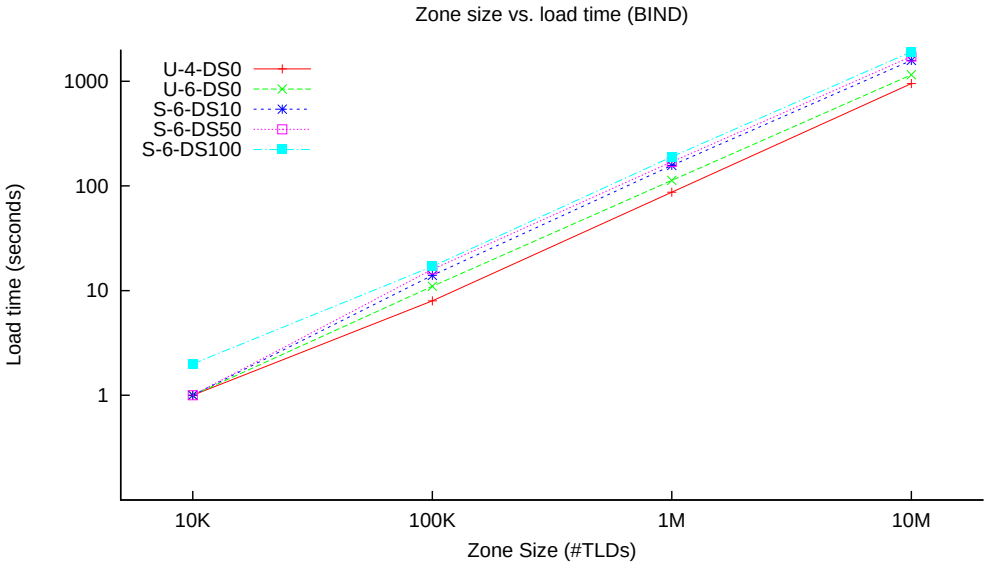

Figure 4.1: Zone size vs. load time – BIND.

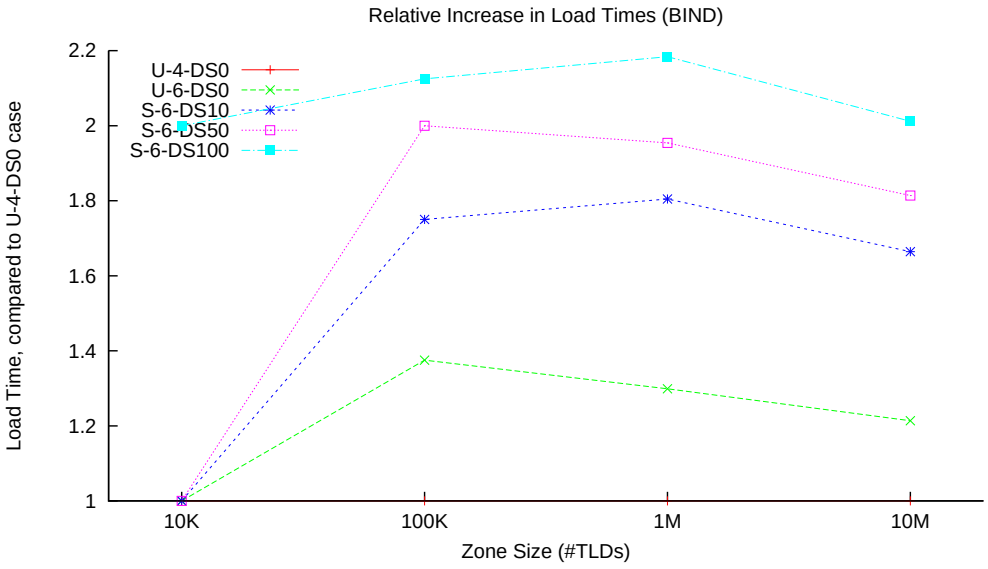

Figure 4.2: Relative change in load time – BIND.

	Zone Size (#TLDs)				
Zone Type	1K	10K	100K	1M	10M
U-4-DS0	<1	<1	8	90	1012
U-6-DS0	<1	<1	11	122	1240
S-6-DS10	<1	2	16	168	N/A
S-6-DS50	<1	2	18	203	N/A
S-6-DS100	<1	2	18	200	N/A

Table 4.2: Reload time (in seconds) vs. zone size – BIND. Note: BIND was unable to reload the largest zones on this platform (32 GB RAM).

the 1K TLD zone loads much faster than our measurement resolution, those values are not shown on this plot.

Figure 4.2 shows the relative change in load times compared to the U-4-DS0 case.

The measurements for BIND's reload tests are shown in Table 4.2. In general BIND takes slightly longer to reload a zone than to load it at startup. For the three largest zones, the reload procedure required more memory than was available on the server (32 GB). This data is shown graphically in Figure 4.3.

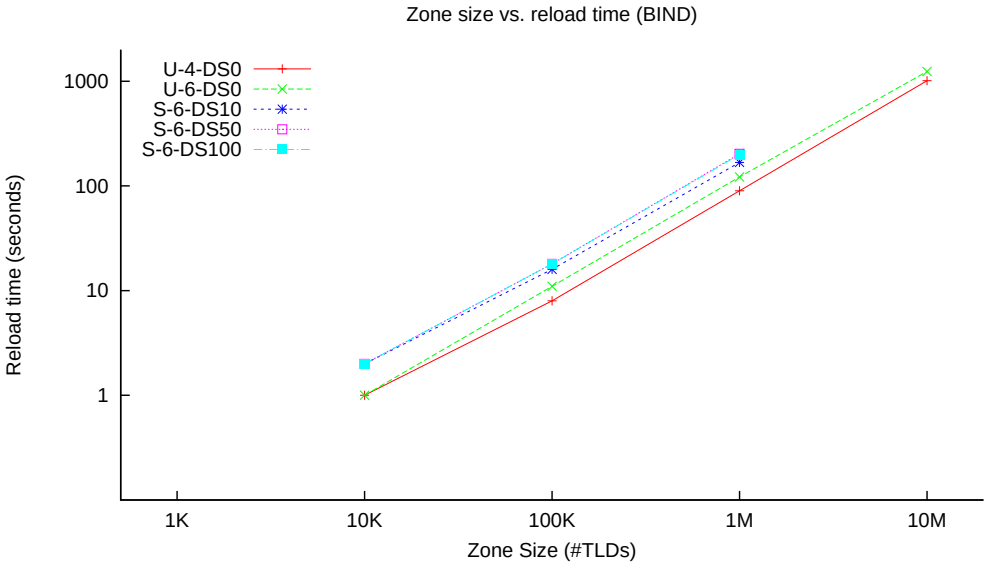

Figure 4.3: Zone size vs. reload time – BIND.

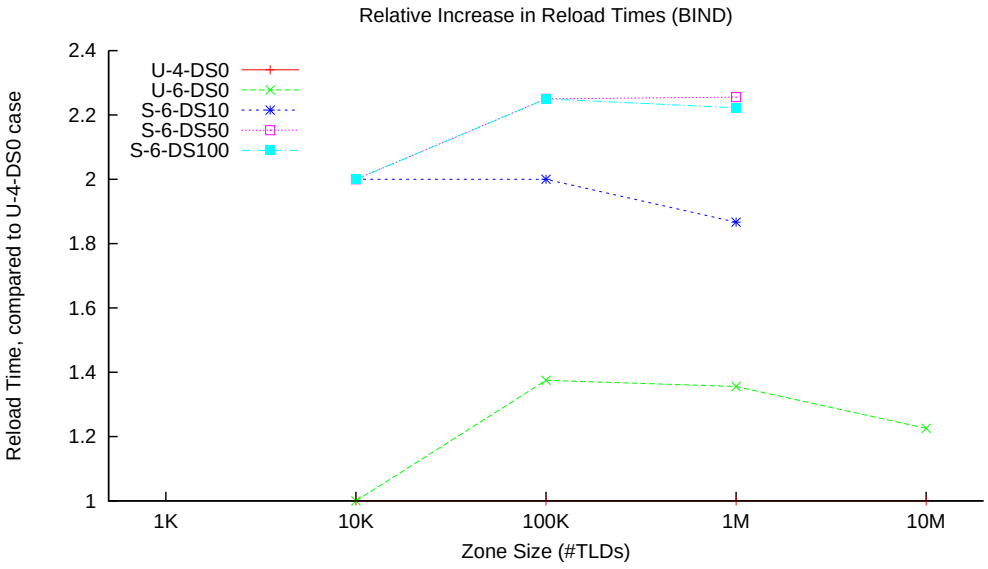

Figure 4.4: Relative change in reload time – BIND.

	Zone Size (#TLDs)				
Zone Type	1K	10K	100K	1M	10M
U-4-DS0	<1	2	13	147	1601
U-6-DS0	<1	2	15	173	1763
S-6-DS10	<1	2	18	197	N/A
S-6-DS50	<1	3	19	210	N/A
S-6-DS100	<1	3	21	227	N/A

Table 4.3: Load time (in seconds) vs. zone size – NSD. Note that NSD was unable to load the signed 10 million TLD zone files on this platform (32 GB RAM).

	Zone Size (#TLDs)				
Zone Type	1K	10K	100K	1M	10M
U-4-DS0	<1	2	14	147	1603
U-6-DS0	<1	2	16	175	1778
S-6-DS10	<1	2	18	203	N/A
S-6-DS50	<1	2	21	211	N/A
S-6-DS100	<1	3	22	231	N/A

Table 4.4: Reload time (in seconds) vs. zone size – NSD. Note that tests of the larger zones exhausted available memory.

4.3 NSD Results

The raw NSD load and reload times are shown in Tables 4.3 and 4.4. Note that NSD uses a two-step process to load zones. First the zone is compiled into a binary format; then it is loaded into memory. The measurements included here represent the amount of time required to run the zone compiler as well as the time required to load the compiled zone. We believe it makes sense to combine both times as this represents the more operationally typical case. The compile time accounts for around 80% of the total time required for NSD to load a zone.

As with the BIND results, we do not report times less than one second. Where observations fall below this minimum resolution, the results are con-

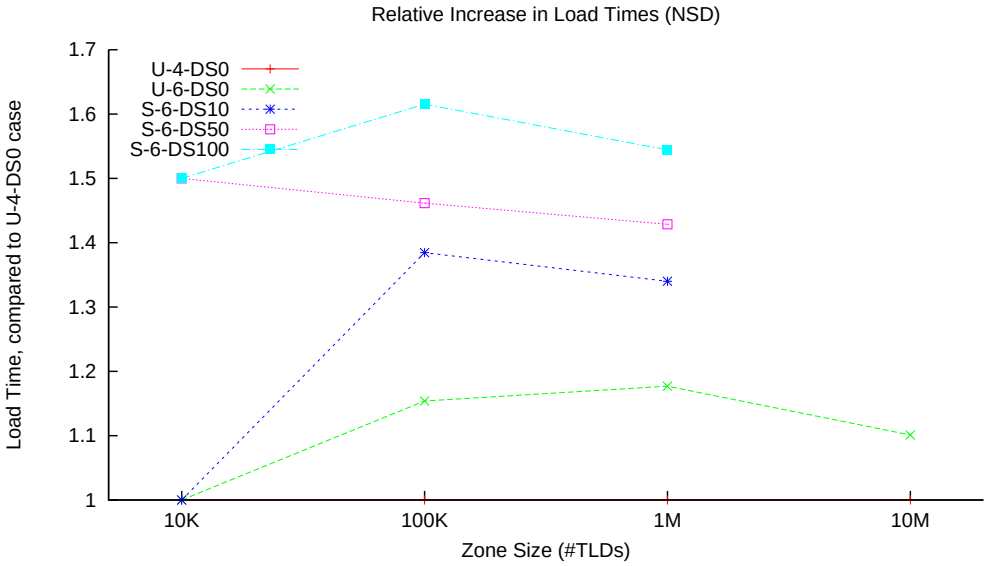

Figure 4.5: Relative change in load time – NSD.

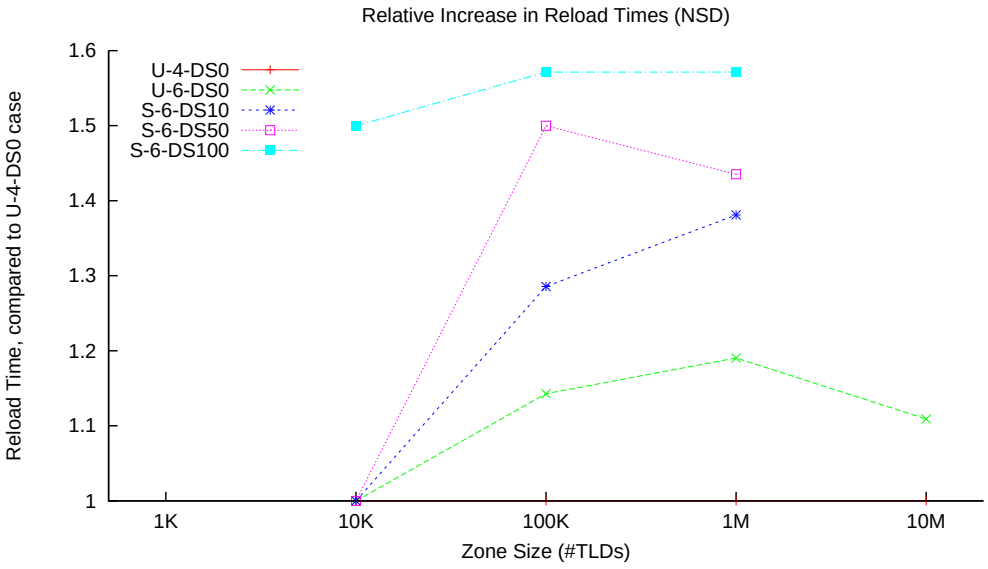

Figure 4.6: Relative change in reload time – NSD.

sidered inconsequential, as from an operational perspective they are practically instantaneous.

Figures 4.5 and 4.6 show the relative change in NSD load and reload times compared to the simple U-4-DS0 cases.

Chapter 5

Impact on Remote Node Bandwidth Utilization

This task examines the impact of an increased number of TLDs in the root zone on the amount of bandwidth required to support transferring the zone contents via AXFR and updating the zone contents via IXFR. We were asked to consider the following:

> *What are the remote node bandwidth requirements necessary to support an increased number of TLDs using AXFR? What are the same requirements using IXFR with two trials, one in which 1% of the entries are being added, deleted, and changed, and the second in which 10% of the entries are being added, deleted, and changed over a 24 hour period? Key variables to examine are:*
>
> a. *Addition of IPv6 addresses for all name servers in the root zone*
> b. *Whether the zone is DNSSEC-signed (using anticipated root DNSSEC keys and key sizes)*
> c. *Percentage of TLDs that have DS RRs with at least 10%, 50%, and 100% considered.*

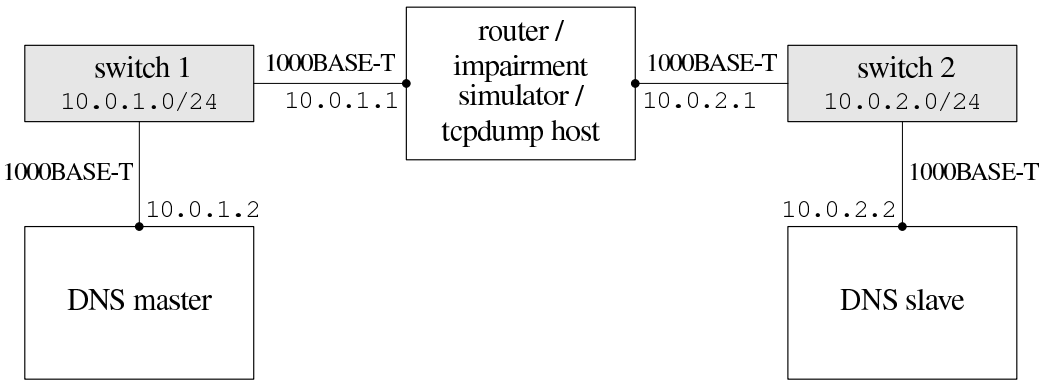

Figure 5.1: Setup for measuring AXFR/IXFR bandwidth.

5.1 Setup

Figure 5.1 shows the setup used for conducting AXFR and IXFR testing. NSD and BIND were installed and configured on two testbed hosts. One server was configured to act as a DNS master for the root zone, the other as its sole slave. A third host was configured to act as a router between the master and slave hosts. tcpdump and NIST Net[5] (a network emulation package) were installed on the router.

A test harness for measuring AXFR transactions was developed in Perl and Bourne shell to perform the following actions:

1. Initialize the setup, removing any saved zone data on the slave.
2. Select root zone to load.
3. Optionally, set network impairments on the router.
4. Start NSD or BIND on the master.
5. Start tcpdump on the router to capture traffic between the master and slave.
6. Start NSD or BIND on the slave, initiating a zone transfer.

7. Repeatedly query the slave for a marker resource record at the end of the zone.
8. Upon detecting the presence of the record, terminate the tcpdump instance.
9. Record the duration of the transfer.
10. Read the pcap file output left by tcpdump and determine the amount of data exchanged during the AXFR (excluding Ethernet headers).

A slightly different test harness for measuring IXFR transactions was developed to perform the following actions:

1. Generate files of incremental updates for each zone. containing data with characteristics similar to the contents of the zone. Each batch of updates contains 10% additions, 10% deletions, and 80% modifications, approximating typical activity in a large TLD.[1]
2. Initialize the setup, removing any saved zone data or journal files on the master and slave.
3. Select root zone to load.
4. Start BIND on the master.[2]
5. Start NSD or BIND on the slave, initiating a zone transfer.
6. Repeatedly query the slave for a marker resource record at the end of the zone.
7. Upon detecting the presence of the record, start tcpdump on the router to capture traffic between the master and slave.
8. Send batches of updates to the BIND master, one at a time, using dynamic DNS (DDNS), triggering an IXFR transfer. Wait until the slave answers for last update before starting the next update.

[1] Additions and deletions tend to be a smaller proportion of overall activity in registries without a redemption grace period; however, we believe this is a reasonable approximation.

[2] We only tested BIND as the IXFR master. While NSD can forward IXFR updates, it cannot generate them on its own.

9. Upon completion of the last update, terminate the tcpdump instance.

10. Read the pcap file output left by tcpdump and determine the amount of data exchanged during the IXFRs (excluding Ethernet headers).

Note that zones larger than the 1M TLDs were not tested for this task. This is because only one host was available that could load zones of this size, and testing AXFR/IXFR requires two such hosts.

Note also that for IXFR testing, a version of BIND was used in which automatic zone re-signing had been disabled, as this created IXFR traffic that interfered with testing. This had to be done at compile-time as there is no reliable configuration option to disable re-signing in BIND 9.6.0-P1 if dynamic updates are enabled.

5.2 AXFR Results

The raw data for AXFR tests is shown in Table 5.1 as the amount of data transferred (in megabytes) for different zone types, sizes, and master/slave configurations. For example, it takes 384 MB to transfer the U-4-DS0 zone with 1,000,000 TLDs between two BIND servers.

One interesting aspect of this data is that when NSD is the master, the AXFR size is significantly (20−30%) smaller. This is because NSD uses message compression as described in Section 4.1.4 of RFC 1035[10] when sending AXFR data, whereas BIND does not.

As expected, AXFR size was proportional to zone size. That is, for a given zone type, the AXFR size increases linearly with the number of TLDs in the zone.

In Figure 5.2 we plot the AXFR size compared to the on-disk zone file size. This offers a comparison between using AXFR and, say, *rsync* to transfer a zone file. The four clusters of points in the graph represent different number of TLDs (1K, 10K, 100K, 1M). Within each cluster are four lines of five points each. The points represent different zone types (S-6-DS10, etc). The lines of different colors represent the four possible master/slave combinations.

The Y-axis in this graph shows the size of an AXFR compared to the size of

Zone Type	Zone Size 1K	10K	100K	1M
BIND→BIND				
U-4-DS0	0.4	3.9	38.6	384.8
U-6-DS0	0.5	6.0	60.5	591.9
S-6-DS10	0.7	8.5	85.2	848.5
S-6-DS50	0.9	9.7	95.0	966.5
S-6-DS100	1.0	11.1	109.8	1112.8
NSD→NSD				
U-4-DS0	0.3	3.0	31.8	315.7
U-6-DS0	0.4	4.6	46.3	419.6
S-6-DS10	0.6	6.7	63.7	625.4
S-6-DS50	0.8	7.2	72.7	715.9
S-6-DS100	0.8	8.3	77.4	754.4
BIND→NSD				
U-4-DS0	0.4	3.9	39.3	388.5
U-6-DS0	0.5	6.1	61.8	600.4
S-6-DS10	0.8	8.6	87.3	872.9
S-6-DS50	0.9	9.8	98.8	988.2
S-6-DS100	1.0	11.2	111.4	1128.3
NSD→BIND				
U-4-DS0	0.3	3.2	31.5	310.3
U-6-DS0	0.4	4.7	47.7	469.8
S-6-DS10	0.6	7.0	69.0	679.0
S-6-DS50	0.8	8.1	73.3	780.6
S-6-DS100	0.9	9.3	92.6	911.8

Table 5.1: Amount of traffic (in megabytes) sent from master to slave during a standard zone transfer for different zone types and sizes.

the on-disk zone file. For example, most of the U-4-DS0 zones with BIND as the master are near the 100% line, indicating that the AXFR size is about the same as the file size. For NSD, the AXFR size is less—closer to 80%. As DS records are added to the zones, the ratio increases significantly. For BIND, these zones have AXFR sizes about 150% of the file size.

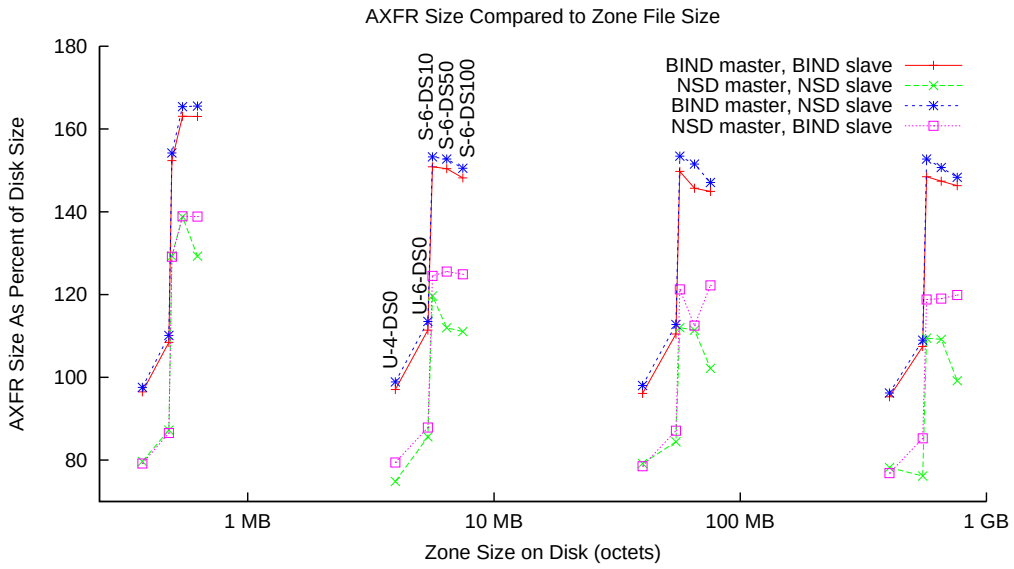

Figure 5.2: AXFR sizes compared to zone file size on disk. The X-axis is scaled logarithmically to show the 1K, 10K, 100K, and 1M TLD cases.

5.3 IXFR Results

Two IXFR scenarios were simulated: one where 10% of the zone contents were replaced in a 24 hour period, and one where 1% of the zone contents were replaced in the same period. Tables 5.2 and 5.3 show the amount of data transferred and the total duration of the transfers for the 10% case. Tables 5.4 and 5.5 show the same information for the 1% case.

We evaluated two different update frequencies: once every 5 minutes, and once every hour, as these represented two possible but distinct update strategies. We evaluated the unsigned (U-4-DS0) and fully populated signed (S-6-DS100) zones only, as we the believe the intermediate cases would be of little interest.

The times given should be taken as indicative only, as they are more representative of BIND's performance as a combined DDNS/IXFR server (and zone signer) than they are of the duration of the IXFRs. In fact, the times became so meaningless for the larger signed zones that we have omitted them

Zone Type	Zone Size 1K	10K	100K	1M
BIND→BIND, 5min updates				
U-4-DS0	0.7	0.9	4.9	57.0
S-6-DS100	0.6	3.1	22.9	198.1
BIND→BIND, 1hour updates				
U-4-DS0	0.1	0.4	4.6	53.2
S-6-DS100	0.2	2.3	24.5	246.4
BIND→NSD, 5min updates				
U-4-DS0	0.5	0.8	4.6	53.0
S-6-DS100	0.7	3.0	22.4	213.0
BIND→NSD, 1hour updates				
U-4-DS0	0.1	0.4	4.6	53.9
S-6-DS100	0.2	2.3	24.2	234.3

Table 5.2: Amount of traffic (in megabytes) sent from master to slave during IXFR transfers changing 10% of the zone contents for different zone types and sizes.

altogether.

For unsigned zones, the bandwidth required to update 10% of the zone via multiple IXFR transactions worked out to be in the range of 12%–15% of the bandwidth required to transfer the zone itself via AXFR. For example, the bandwidth required to update 10% of the unsigned 1M zone (U-4-DS0, BIND→BIND, 5min updates) was 57 MB, which is 14.8% of 385 MB, the bandwidth required to transfer the zone. In general, the larger the zone, the smaller the percentage was of the incremental update bandwidth relative to the AXFR bandwidth.

For signed zones, the bandwidth required to update 10% of the zone was closer to 20% of the zone's AXFR bandwidth.

One issue we encountered was the speed of BIND when sending DDNS updates to a DNSSEC-signed zone. Because BIND signs each updated resource record, there is a significant overhead when compared with updates to an unsigned zone. The update speed was especially slow for the S-6-DS100-1M zone, where updates could be sent no faster than three resource records per second. Also, DDNS updates must be sent in relatively small transactions (around 100 RRs per transaction) to avoid SERVFAIL errors. Strictly

Zone Type	Zone Size 1K	10K	100K	1M
BIND→BIND, 5min updates				
U-4-DS0	1415	1435	1441	3313
S-6-DS100	1411	1436	*	*
BIND→BIND, 1hour updates				
U-4-DS0	115	121	607	723
S-6-DS100	116	120	*	*
BIND→NSD, 5min updates				
U-4-DS0	2523	2590	2632	2678
S-6-DS100	60	2590	*	*
BIND→NSD, 1hour updates				
U-4-DS0	210	220	690	883
S-6-DS100	208	228	*	*

Table 5.3: Duration (in seconds) of aggregate IXFR transfers while changing 10% of the zone contents for different zone types and sizes. Times for the larger signed zones have been omitted.

Zone Type	Zone Size 1K	10K	100K	1M
BIND→BIND, 5min updates				
U-4-DS0	0.16	0.16	0.95	4.69
S-6-DS100	0.64	0.67	3.13	28.20
BIND→BIND, 1hour updates				
U-4-DS0	0.003	0.08	0.45	5.35
S-6-DS100	0.004	0.25	2.30	25.00
BIND→NSD, 5min updates				
U-4-DS0	0.15	0.16	0.78	4.55
S-6-DS100	0.68	0.68	3.22	24.79
BIND→NSD, 1hour updates				
U-4-DS0	0.002	0.07	0.45	5.35
S-6-DS100	0.003	0.25	2.15	25.84

Table 5.4: Amount of traffic (in megabytes) sent from master to slave during IXFR transfers changing 1% of the zone contents for different zone types and sizes.

Zone Type	Zone Size 1K	10K	100K	1M
BIND→BIND, 5min updates				
U-4-DS0	61	62	1435	1441
S-6-DS100	64	79	*	*
BIND→BIND, 1hour updates				
U-4-DS0	5	115	120	121
S-6-DS100	5	115	*	*
BIND→NSD, 5min updates				
U-4-DS0	58	57	2590	2658
S-6-DS100	60	76	*	*
BIND→NSD, 1hour updates				
U-4-DS0	5	208	222	221
S-6-DS100	5	209	*	*

Table 5.5: Duration (in seconds) of aggregate IXFR transfers while changing 1% of the zone contents for different zone types and sizes. Times for the larger signed zones have been omitted.

speaking, these considerations are out-of-scope, as the task involves measuring IXFR characteristics only, but we believe they are worth mentioning.

5.4 Bandwidth and Latency

The previous results show the total amount of data sent during zone transfers; however any analysis of required bandwidth must take into account the latency of the path between two nodes. In practice, latency will be the principal limiting factor on the speed with which a zone may be transferred.

To simulate the effect of network latency, we set up NIST Net on the router (Figure 5.1). NIST Net permits a user to specify the desired level of several impairments: latency, packet loss, duplicate packets, and bandwidth, as well as a few modifiers. We performed the AXFR and IXFR transactions from the previous sections with varying levels of latency and packet loss to simulate the behavior of these operations over long distances and imperfect links.

Figure 5.3 shows the effective bandwidth imposed by various latencies. In effect, a 30ms round-trip latency imposes an upper limit on bandwidth of

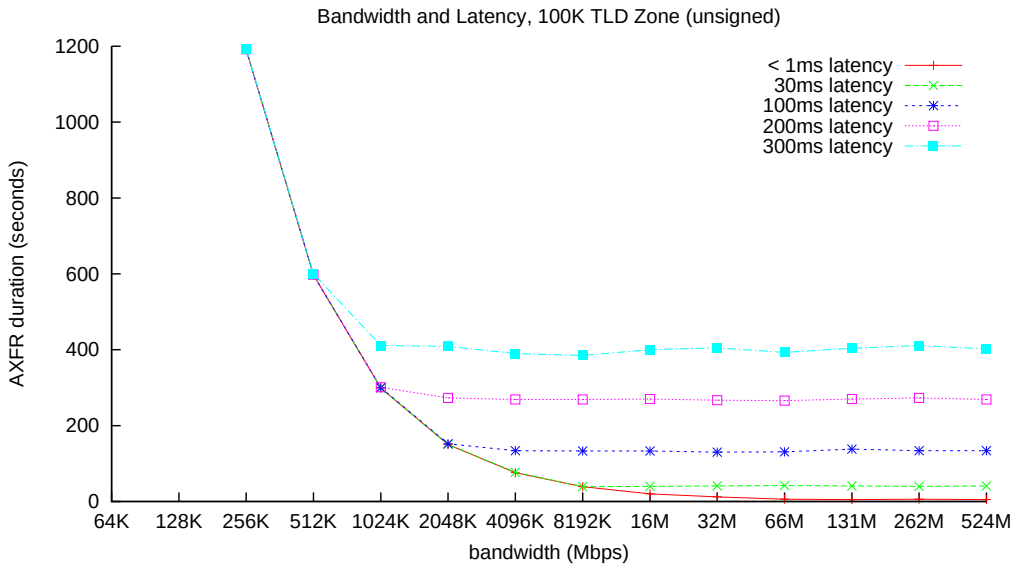

Figure 5.3: Bandwidth and Latency (100K U-4-DS0 zone, BIND→BIND).

8.2 Mbps between two nodes. A 100ms latency imposes a 2 Mbps limit, 200ms a limit of 1 Mbps, and 300ms a limit of 800 Kbps. In other words, if you are trying to send a file over a link with 300ms latency, whether you have 1 Mbps of bandwidth available or 100 Mbps you will be limited to 800 Kbps throughput.

5.5 Effects of Latency and Packet Loss

Packet loss has an additional deleterious effect. Figure 5.4 shows the duration of zone transfers for the unsigned 100K TLD (U-4-100) zone at three different latencies: 30ms, 100ms, and 200ms, and also shows the amount of time they took at the same latencies when a 1%, 2%, 3%, and then 4% packet loss was imposed on top of the latency. Counterintuitively, a 1% packet loss rate caused transfer times to more than double while 2% added only a small additional time penalty. Figure 5.5 shows the relative increase in AXFR duration compared to the 0% loss case.

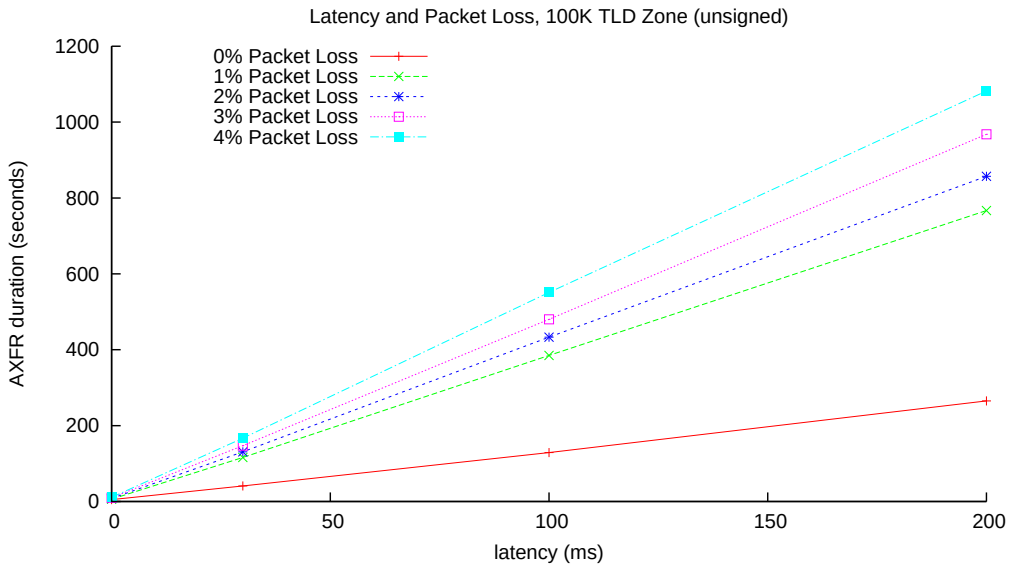

Figure 5.4: Effect of latency and packet loss on AXFR duration (100K U-4-DS0 zone, BIND→BIND).

5.6 Caveat About NIST Net

NIST Net uses a Gaussian distribution to represent variations in latency, i.e., the distribution of delays imposed on a traffic flow. Latency observed on the Internet rarely has such pure characteristics. Figure 5.6 shows the difference between AXFR transactions across NIST Net (in red) and AXFR transactions between two widely separated hosts (green)[3]. Both distributions have a median of around 245 milliseconds, but otherwise have very different characteristics. In particular, the real-world distribution has a long tail that is indicative of lower throughput.

[3]DNS-OARC gratefully acknowledges NZRS for use of a dedicated host for this measurement.

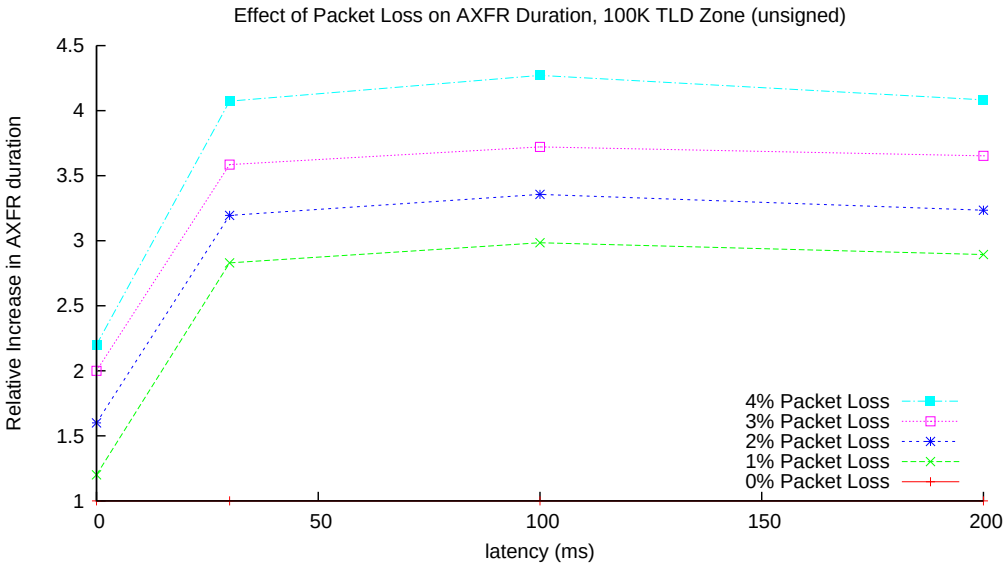

Figure 5.5: This plot shows the increase in AXFR duration compared to the 0% loss case.

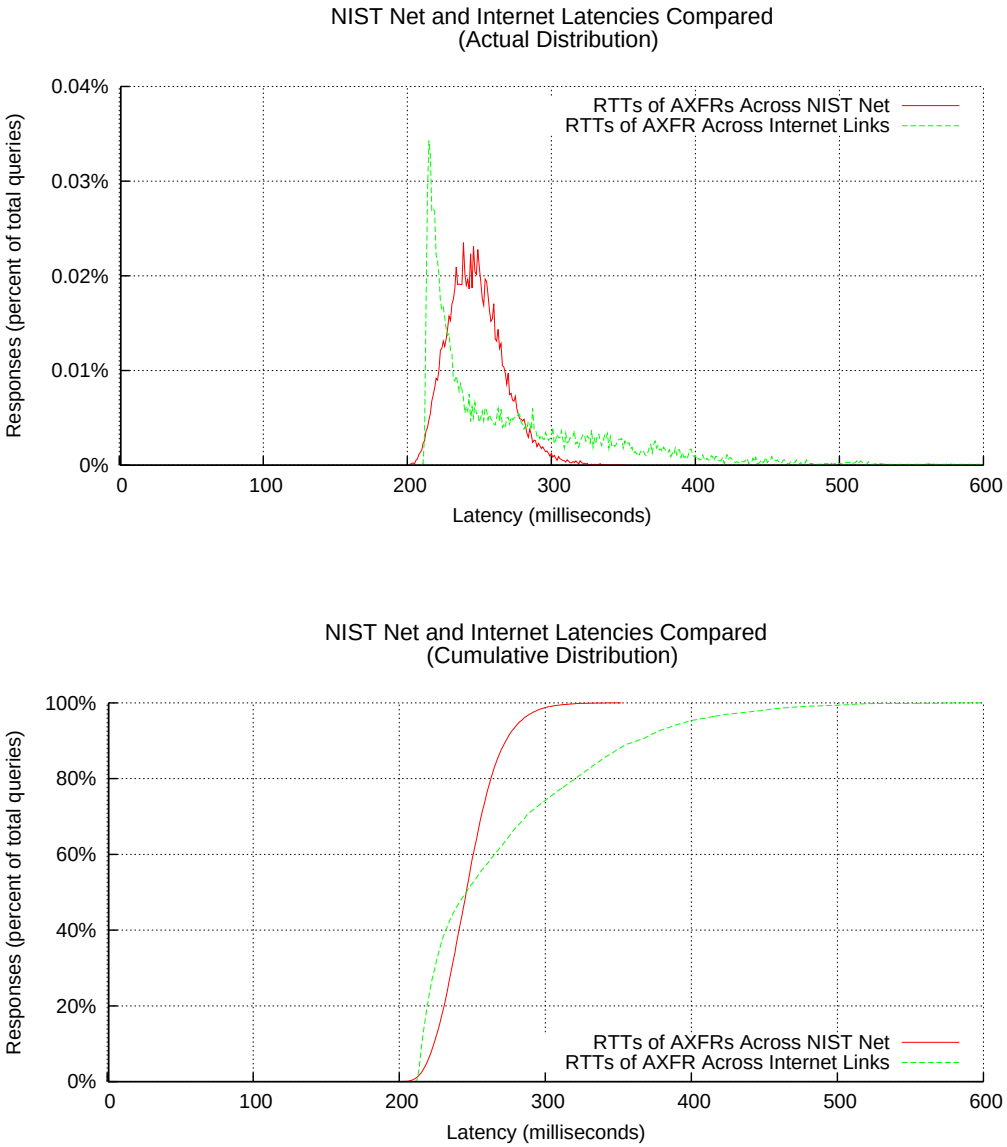

Figure 5.6: NIST Net and Internet latencies compared.

Chapter 6

Impact on TCP Usage

This task examines the impact of DNSSEC on TCP usage at the root servers. We were asked to consider the following:

> *What level of increased TCP usage will result with the addition of DNSSEC (using the anticipated number and sizes of key) and/or IPv6 records for all name servers in the root zone?*

6.1 Overview

To obtain a comprehensive picture of the potential impact of TCP usage on a signed root zone, we examined several different aspects of the issue:

- We surveyed recent levels of TCP-based DNS activity on the root servers collected during the 2009 DITL Data Collection Project.
- We replayed UDP-based DNS queries from DITL to a version of the current root zone signed using the ICANN testbed parameters to determine what percentage of queries would result in truncated replies.
- We examined traffic from name servers for several ccTLDs with signed zones to determine what percentage of truncated replies resulted in subsequent queries over TCP.

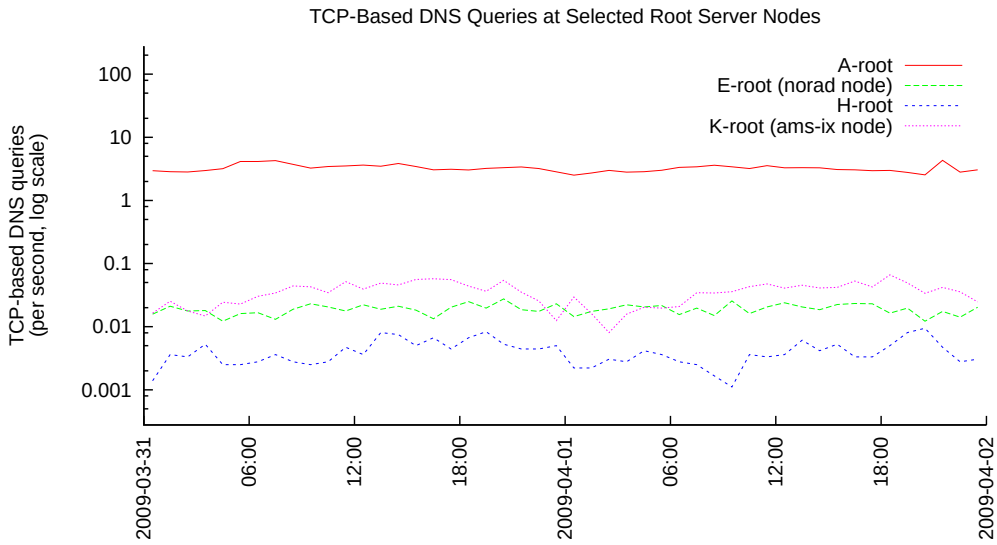

Figure 6.1: TCP-based DNS queries at selected root server nodes.

- Finally, we examined data for the `.org` TLD from before and after DNSSEC was deployed to identify any patterns that might have implications for a signed root.

6.1.1 Current TCP Activity

Current DNS usage over TCP is miniscule. The notable exception is A-root, which receives thousands of dynamic DNS update and TKEY key exchange attempts every hour via TCP. A-root is specified as the primary server in the root zone's SOA resource record, making it a target for misconfigured hosts.

Figures 6.1 and 6.2 show DNS queries received via TCP at several root server instances. Queries were identified by inspecting TCP packets for DNS messages with valid DNS headers and with the QR-bit set to zero.

Figure 6.1 shows the number of TCP-based DNS queries received per second. A-root received around three per second, while the other instances received

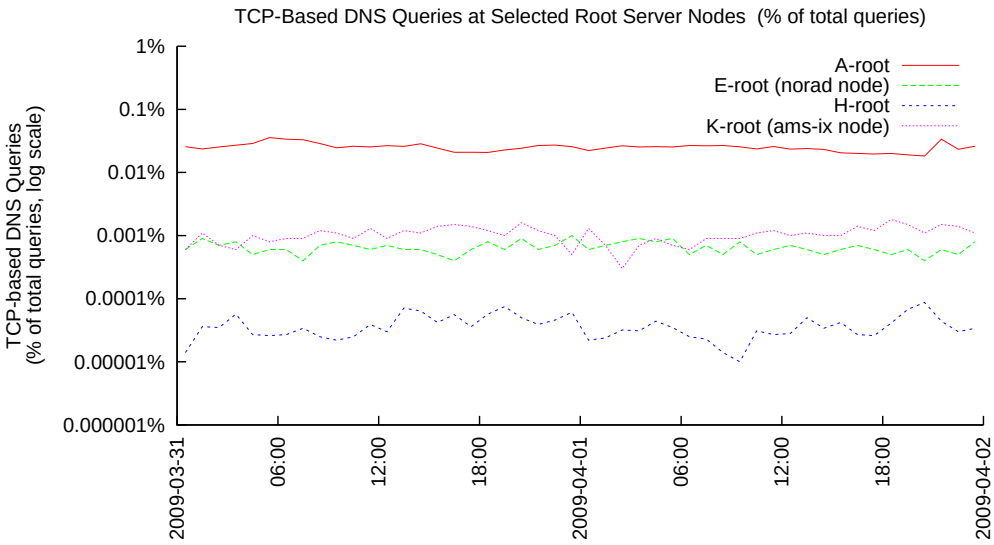

Figure 6.2: TCP-based DNS queries at selected root server nodes (as a percentage of all queries to that node).

considerably fewer. (Data for L-root was unavailable as the L-root DITL data contained UDP datagrams only.)

Note that using Figure 6.1 to compare root servers is relatively meaningless: the rates are for individual nodes only, and do not reflect the total number of TCP-based DNS queries received by each root server as a whole. More meaningful comparisons can be made by looking at the *ratio* of TCP-based queries to total queries for each server instance (Figure 6.2). Here, differences between root servers are still apparent, with A-root receiving around 0.025% of DNS queries via TCP while others receive around 0.001% or less.

A closer look at the handful of queries that are transmitted over TCP reveals that none are in response to truncated replies. A significant proportion contain nonsense; we came across a number of examples like these:

```
$ tshark -r tcp.pcap 'tcp && dns && ip.addr == 74.125.0.0/16'
1238  74.578687  74.125.16.3 -> 193.0.14.129 DNS Standard query A <Root>
4344 248.633254 74.125.16.66 -> 193.0.14.129 DNS Standard query A misli
4908 281.727130 74.125.16.65 -> 193.0.14.129 DNS Standard query A <Root>
5296 308.855164  74.125.16.3 -> 193.0.14.129 DNS Standard query A oh
```

```
6161 356.254802  74.125.16.1 -> 193.0.14.129 DNS Standard query A <Root>
8332 481.642632 74.125.16.65 -> 193.0.14.129 DNS Standard query A hi
8495 492.385850 74.125.16.65 -> 193.0.14.129 DNS Standard query A hi
8602 499.311380 74.125.16.66 -> 193.0.14.129 DNS Standard query A really
9112 528.542159 74.125.16.66 -> 193.0.14.129 DNS Standard query A really
9752 567.951381 74.125.16.66 -> 193.0.14.129 DNS Standard query A really
17056 1009.097723  74.125.16.1 -> 193.0.14.129 DNS Standard query A ok
17262 1024.872518  74.125.16.1 -> 193.0.14.129 DNS Standard query A ok
17497 1040.121022  74.125.16.1 -> 193.0.14.129 DNS Standard query A ok
17605 1047.974485  74.125.16.1 -> 193.0.14.129 DNS Standard query A ok
18539 1108.659565 74.125.16.66 -> 193.0.14.129 DNS Standard query A whatever
21353 1279.307606  74.125.16.1 -> 193.0.14.129 DNS Standard query A aha
26454 1592.417938 74.125.16.66 -> 193.0.14.129 DNS Standard query A <Root>
29593 1793.150752 74.125.16.66 -> 193.0.14.129 DNS Standard query A thing
33618 2036.589425  74.125.16.2 -> 193.0.14.129 DNS Standard query A insomma
33660 2039.464841  74.125.16.2 -> 193.0.14.129 DNS Standard query A insomma
33717 2041.883686  74.125.16.2 -> 193.0.14.129 DNS Standard query A insomma
```

We were unsurprised that current DNS activity over TCP was low at the root servers. However, we were very surprised to discover something else: the root servers were all experiencing high rates of TCP session establishment on port 53 (Figure 6.3). This activity initially escaped detection as these connections carried no DNS messages.

The connection rates were measured by counting the number of TCP packets with the FIN and ACK bits set and dividing by two; this gives a close approximation for the number of cleanly-terminating TCP sessions.[1] The average connection rate on A-root alone was 80 per second.

Most of these connections were no more than a handshake followed immediately by a client-initiated disconnect. The TCP packets contained no payloads, so no data was exchanged at the application level. Figure 6.4 shows that, depending on the root server, these "no-op" connections outnumbered connections bearing DNS queries by between 25:1 (for A-root) to 1000:1 (for H-root).

The connections came from a broad distribution of hosts. A cursory investigation suggests that many of the source addresses lie in the network allocations of broadband providers, so it would be reasonable to suspect the involvement of compromised hosts.

To summarize the current TCP activity at the root servers:

[1]The actual connection rate may be slightly higher, though a count of packets with the SYN and ACK bits set shows it would be 10% higher at most.

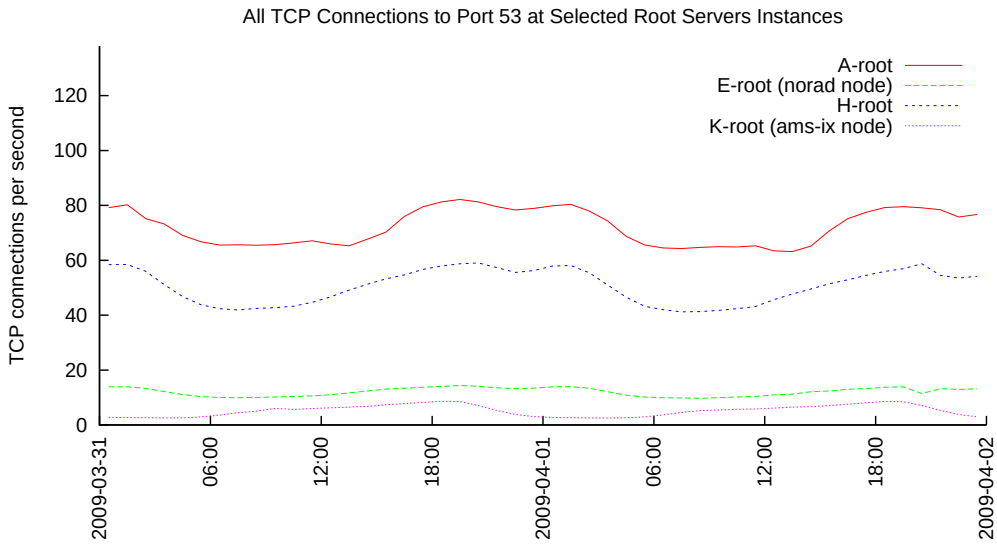

Figure 6.3: All TCP connections (with and without DNS payloads) to port 53 at selected root server nodes.

- The observed rate of DNS queries over TCP is extremely low on all root servers with the exception of A-root.
- There was a high number of "junk" TCP connections to the root servers during the DITL collection period.

6.2 Prevalence of Truncated Replies

It is well-understood that a signed root zone will result in an increase in TCP-based DNS queries. Responses to queries with the DO-bit set will be significantly larger, especially for RCODE = 3 (NXDOMAIN) replies. To attempt to forecast the possible increase, we set up a name server with a signed instance of the current root zone and replayed traffic from a number of root server nodes to see how many truncated replies it returned.

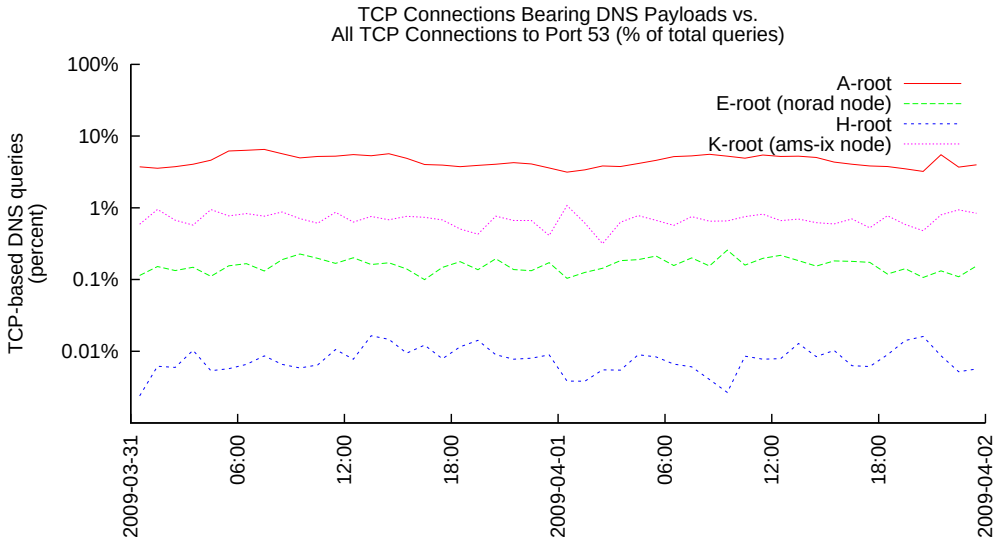

Figure 6.4: TCP connections bearing DNS payloads at selected root server nodes (as percentage of all TCP connections to port 53 on that node).

6.2.1 Setup

The setup for replaying DNS queries was similar to the one used in Task 2 (Figure 3.1) to measure query latency. A version of the current root zone was signed using the ICANN testbed parameters and loaded into NSD. Pcap files were prepared from the DITL data as follows:

- Queries were extracted from the DITL data to pcap files using a utility we wrote named `pcap_extract_queries` to sample every 100th query and also to group data into time slots:

  > `pcap_extract_queries -S 100 -s` ⟨start time⟩ `-e` ⟨end time⟩ `-r` ⟨original DITL pcap⟩ `-w` ⟨new pcap file⟩ ⟨original DITL pcap file⟩ ...

  The result was 48 pcap files each containing one hour of query data from midnight (GMT) March 31$^{\text{st}}$ to midnight April 2$^{\text{nd}}$, 2009.

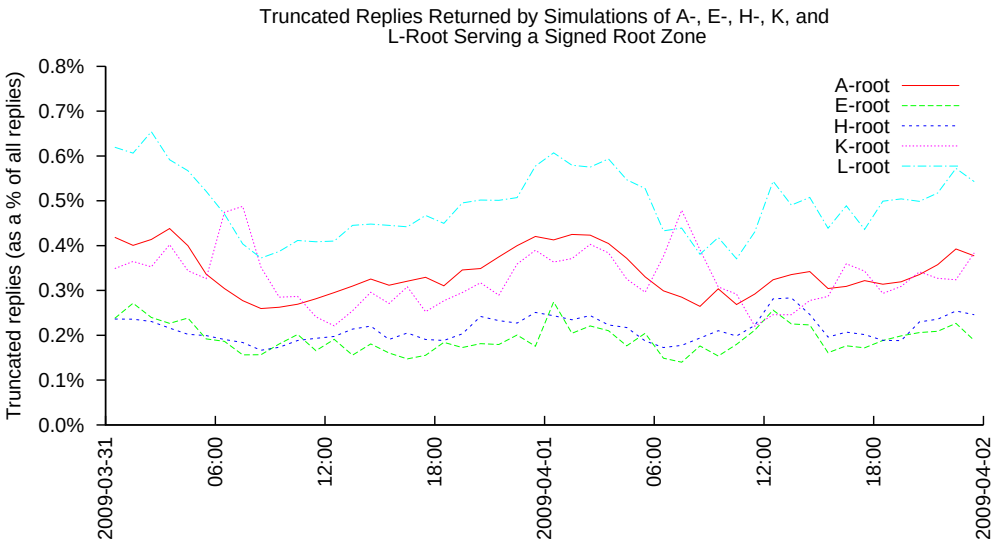

Figure 6.5: Rates of truncated replies from simulations of A-, E-, H-, K-, and L-root serving a signed root zone.

- Ethernet headers were inserted (if pcap type = RAW) or rewritten (if pcap type = EN10MB) with the IP address and MAC addresses of the testbed servers using `pcap_rewrite` (a utility we wrote):

  `pcap_rewrite -e` ⟨dummy source MAC address⟩ `-E` ⟨destination MAC address⟩ `-i` ⟨destination IP address⟩ ⟨pcap_file⟩

The resulting queries were then transmitted using `tcpreplay`:

`tcpreplay -i` ⟨interface⟩ ⟨rewritten pcap_file⟩

The responses were captured using `tcpdump`, and DNS replies with the TC-bit set were tallied.

6.2.2 Results

Figure 6.5 shows the rate of replies with the TC-bit set that would have been returned by instances of A-, E-, H-, K-, and L-root servers had the

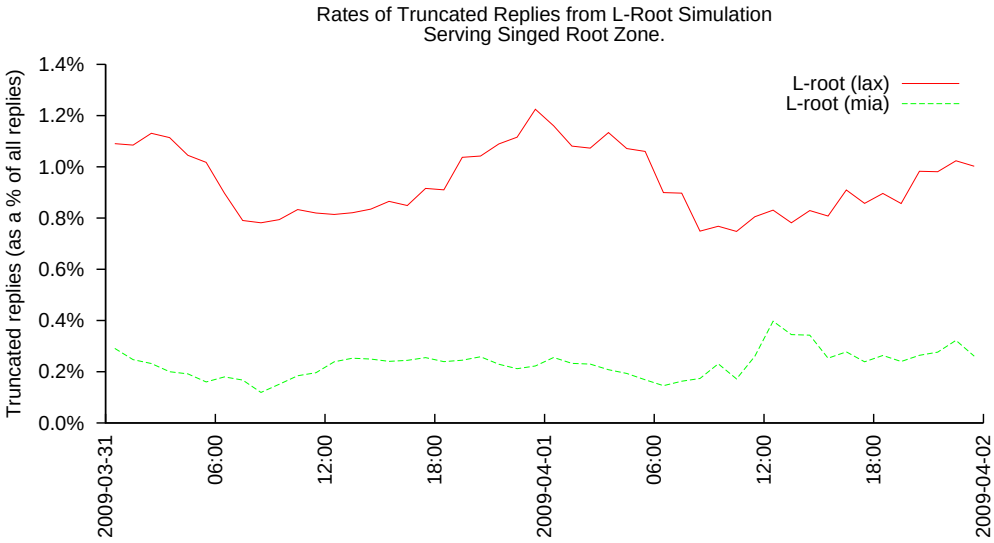

Figure 6.6: Rates of truncated replies returned by simulations of L-root LAX and MIA nodes serving signed root zones.

root zone been signed during the 2009 DITL collection period. In actuality the rates would have varied slightly, as the precise sequence of queries would be altered by a signed root zone. For example, there would have been some (though probably few) queries for the DNSKEY resource records by security-aware resolvers. Nevertheless, the truncation rates are likely to have been substantially similar.

Roughly between 0.2% and 0.6% all queries would have resulted in truncated replies. The truncated replies were in response to two types of queries:

1. Queries with an EDNS0 buffer size of 512 and the DO-bit set.
2. Queries with a QNAME of "." and a QTYPE of ANY.

The first type of query was overwhelmingly the most prevalent: wildcard queries constituted less than 0.01% of queries resulting in truncated replies.

Differences in truncation rates between servers are attributable to differences in the composition of queries reaching the servers. L-root had a surprisingly

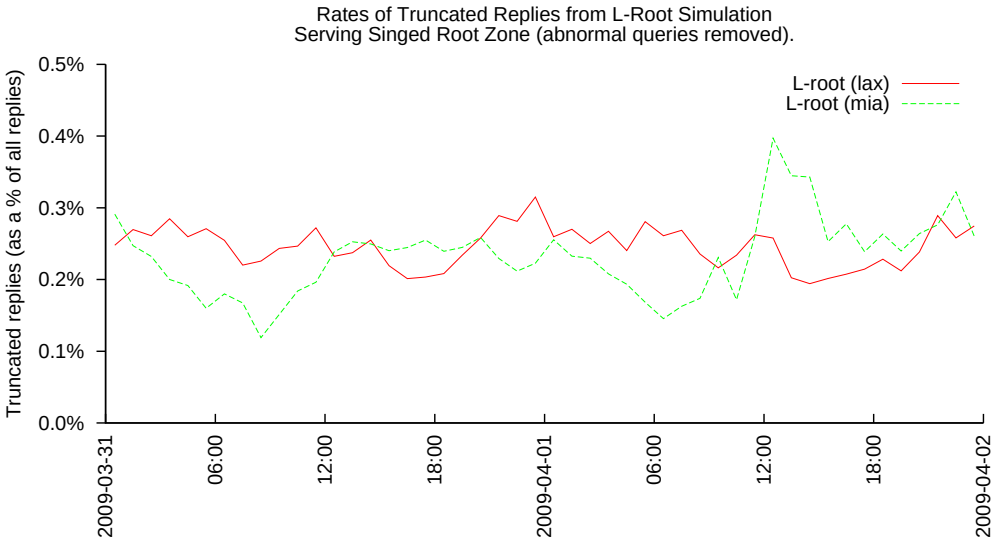

Figure 6.7: Rates of truncated replies returned by simulations of L-root LAX and MIA nodes serving signed root zones, with abnormal queries removed.

high truncation rate relative to the other root servers. On closer examination, we found a big difference between the truncation rate between the Los Angeles (lax) and Miami (mia) nodes (Figure 6.6).

This turned out to be attributable to high rates of queries from 22 hosts in two small netblocks belonging to the same network provider. When these were removed, the difference was markedly reduced (Figure 6.7).

We wanted to see what the impact of a larger zone would be on truncation. We replayed the same A-root queries to an instance of NSD serving the 1M TLD S-6-DS100 zone. The results (Figure 6.8) show that the ratio of truncated replies increases substantially, though not alarmingly. This is primarily due to the larger size of RCODE = 3 (NXDOMAIN) responses due to longer owner names in NSEC resource record included in the reply.

We were also interested in whether there were any differences in truncation behavior between BIND and NSD when presented with the same set of queries. As the plot in Figure 6.9 shows, any difference is virtually indistinguishable.

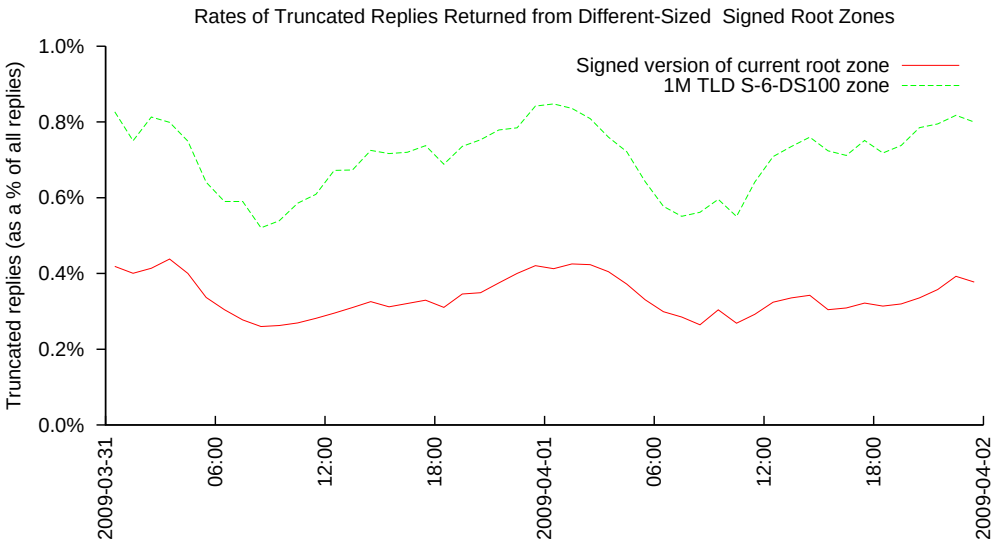

Figure 6.8: Rate of truncated replies returned by simulations of A-root serving small and large signed root zones.

Note that only IPv4 traffic was included in this analysis. This partially due to the unavailability of IPv6 traffic in many of the root server traces. Even when present, the number of queries over IPv6 that resulted in truncated replies was negligible.

6.3 Analysis of Truncated Replies

We were interested in the characteristics of the replies that were truncated. In particular, we wanted to see 1) the sizes of the original replies that were truncated, and 2) the reply types, e.g., whether they were referrals or an name error (RCODE = 3).

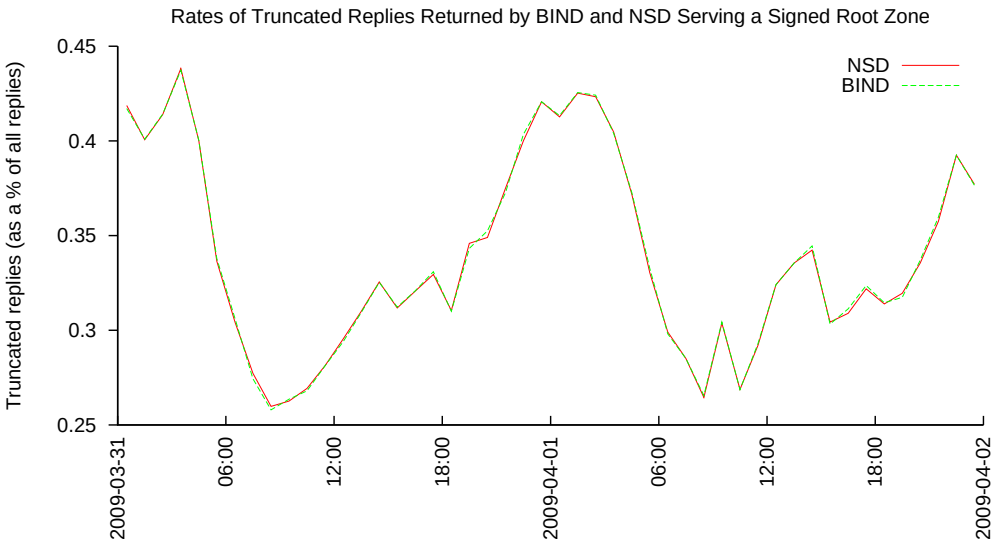

Figure 6.9: Rate of truncated replies returned by BIND and NSD in simulations of A-root serving a signed root zone.

6.3.1 Setup

Using the same DITL pcap files for A-root used in the previous section, we extracted all queries where an EDNS0 buffer of 512 octets was specified and the DO-bit set, changed the EDNS0 size to 65535 (the maximum), and then replayed them to the signed root zone server instance.

- Queries were extracted to pcap files using the `pcap_extract_512do` utility (a utility we wrote):

  `pcap_extract_512do -r` ⟨original DITL pcap⟩ `-w` ⟨new pcap⟩

- Ethernet headers were inserted (if pcap type RAW) or rewritten (if pcap type EN10MB) with the IP address and MAC addresses of the testbed servers using `pcap_rewrite` (a utility we wrote):

  `pcap_rewrite -e` ⟨dummy source MAC address⟩ `-E` ⟨destination MAC address⟩ `-i` ⟨destination IP address⟩ ⟨pcap_file⟩

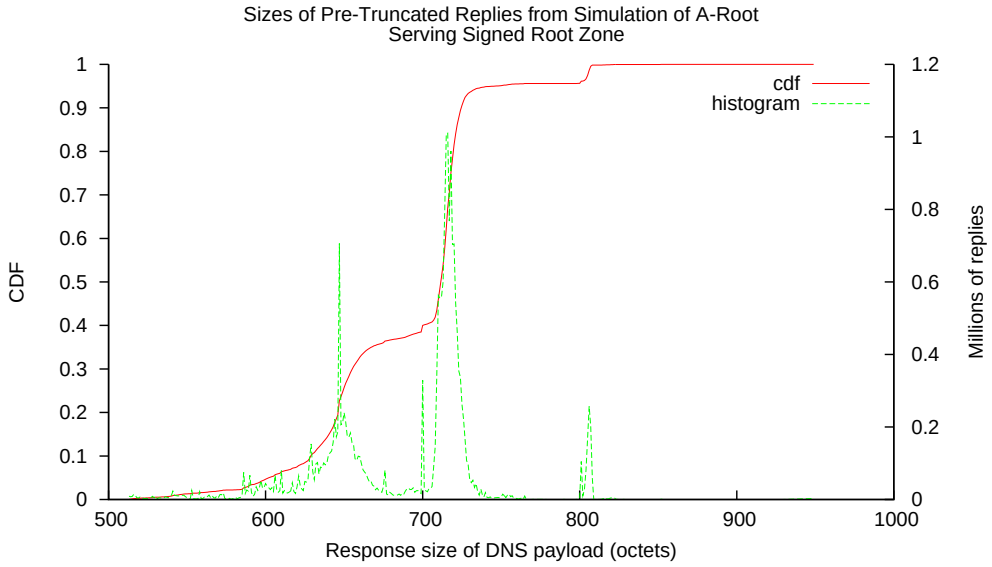

Figure 6.10: Sizes of pre-truncated responses to queries with EDNS buffer size = 512 returned by a simulation of A-root serving a signed root zone.

The resulting queries were transmitted to the server using `tcpreplay`:

`tcpreplay -i` ⟨interface⟩ ⟨rewritten pcap_file⟩

The responses were captured using `tcpdump`.

Figure 6.10 summarizes the results. The response sizes cluster around two major peaks (vertical segments in the CDF). The first peak spikes markedly around 629 octets and corresponds with the size of responses for non-existent names, i.e., RCODE = 3 (NXDOMAIN). The second, more pronounced spike occurs at 716 octets and corresponds with the size of signed referrals containing 13 NS RRs, e.g, for `.com`, `.org` and `.net`.

6.4 Probability of Retries Over TCP

The key operational impact of truncated replies is that they may result in query attempts via TCP; however, truncated replies do not automatically

result in retries over TCP. Popular resolvers will retry, but queries may originate from other sources, e.g., scripts, compromised hosts in a botnet, and poor-quality resolver implementations in embedded devices. To get an indication of what proportion of truncated responses will result in subsequent retries over TCP, we examined DITL data from the name servers for several ccTLDs that have deployed DNSSEC. The data for `.se` (the ccTLD for Sweden) provided the clearest picture.

Figure 6.11 shows the data observed for the `f.ns.se` name server. The green line shows the number of truncated replies returned by `f.ns.se`, and the blue line shows the number of subsequent queries received via TCP with the same QNAME/QTYPE/QCLASS over TCP. It is apparent that the retry rate is high: it averaged 88.11% over the two-day period. This suggests that when the root zone is signed, the number of retries over TCP will be close to the number of truncated replies returned.

One surprising observation is that nearly 90% of queries repeated over TCP differed from the original queries in that the DO-bit was no longer set; the OPT resource record was omitted altogether. The number of retries with the DO-bit set is shown by the magenta line in Figure 6.11. fpdns pointed to older versions of BIND (pre 9.4); recent versions of BIND do not exhibit this behavior.

6.5 `.org` Deployment Experience

The `.org` zone was signed on June 2, 2009. Public Interest Registry, the manager of the `.org` zone, provided pcap files to DNS-OARC of DNS traffic data to the `.org` names servers covering the day that the signed zone was published. We examined this traffic to identify any patterns that might have implications for a signed root zone.

One pattern was immediately obvious: the number of TCP connections jumped from practically zero to a mean of 58 per second. This was 1.2% of all DNS queries (UDP and TCP).

There was another significant change: the number of queries advertising an EDNS payload size of 512 increased sixfold, from a mean of 16 to 104 queries per second. We believe this increase is due to resolvers compensating for

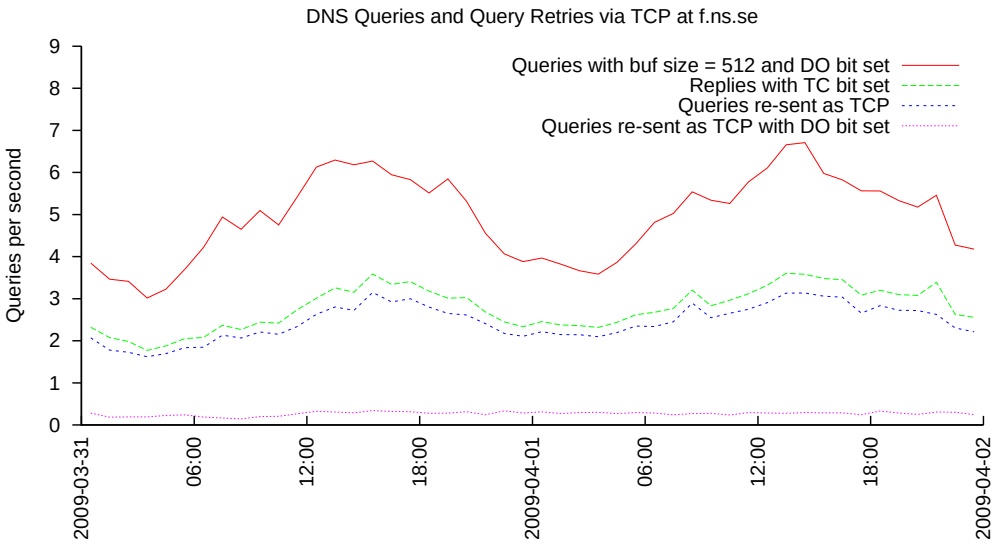

Figure 6.11: Query retries via TCP at `f.nic.se`.

replies lost due to path MTU bottlenecks. Typically these are due to poor-quality firewall implementations that block DNS packets with payloads larger than 512 octets, though they may occur for other reasons. In particular, BIND (versions 9.4.0 and later) resolvers fall back to advertising an EDNS payload size of 512 if the initial EDNS queries fail.[14]

The implication of the increase in queries advertising an EDNS payload size of 512 is that there will be substantially more queries that will result in truncated replies than can be forecast by evaluating current traffic patterns.

6.5.1 Appraisal

Our simulations show that signing the root zone leads to a significant increase in TCP queries at root servers. We know (from DITL and other data, for example) that there is a strong correlation between queries with an EDNS buffer size of 512 octets, responses to those queries with the TC bit set, and a followup query over TCP. Our simulations of A-root show that between 0.4 and 0.8 percent of UDP responses will be truncated when the root zone is

signed. At current traffic levels, this would lead to between 50 and 100 TCP queries per second (in addition to the 10 per second already seen there). Based on data gathered during the signing of the `.org` zone, the actual increase could be even higher than this projection.

Even though all 13 root servers appear to support TCP at this time, an order of magnitude increase in TCP traffic is concerning for a few reasons. First and foremost, perhaps, is that TCP requires additional state on the server. That is, each (non-closed) TCP connection consumes some resources in the server's kernel. One aspect we did not attempt to simulate here is clients that elect to keep a TCP connection open (i.e., for future reuse) after receiving a response.

Additionally, TCP-based servers (applications) must usually work harder to protect against denial-of-service conditions. For example, since each TCP connection requires a separate file descriptor, the application or an intermediate device (e.g., a router) must take steps to prevent file descriptor exhaustion.

Finally, there is some concern that problems will arise due to the highly distributed nature of root DNS servers (i.e., anycast and local load balancers). TCP requires all packets in a TCP flow to arrive at the same server instance. A number of studies [9, 11, 12, 13] have considered the stability of anycast. While we do not have any additional data to add at this time, we do not believe anycast or load balancer instability to be a significant problem for short-lived TCP DNS transactions.

Bibliography

[1] ICANN, *New gTLD Program*. http://www.icann.org/en/topics/new-gtld-program.htm.

[2] Koch, P., *DNS Glue RR Survey and Terminology Clarification*, Work in Progress, November 2007. http://tools.ietf.org/html/draft-koch-dns-glue-clarifications.

[3] procps, http://procps.sourceforge.net/.

[4] pmap, http://www.freebsdsoftware.org/sysutils/pmap.html.

[5] NIST Net, http://www.antd.nist.gov/nistnet/.

[6] DITL 2009 Data Collection Project, https://www.dns-oarc.net/ditl/2009.

[7] E. Osterweil et al., *Availability Problems in the DNSSEC Deployment*, Presented at RIPE 58, May 2009. http://www.ripe.net/ripe/meetings/ripe-58/content/presentations/dnssec-deployment-problems.pdf.

[8] D. Wessels and S. Castro, *DNSSEC, EDNS, and TCP*, Presented at NANOG 46, June 2009. http://www.nanog.org/meetings/nanog46/presentations/Wednesday/wessels_light_N46.pdf.

[9] Z. Liu et al., "Two Days in the Life of the DNS Anycast Root Servers", In *Proceedings of the Passive and Active Measurement (PAM) Conference 2007*: 125–134. Springer Verlag, 2007.

[10] Mockapetris, P., *Domain names - implementation and specification*, STD 13, RFC 1035, November 1987.

[11] M. Larson, *Traffic Source Analysis of the J Root Anycast Instances*, Presented at NANOG 39, November 2006. http://www.nanog.org/meetings/nanog39/presentations/larson.pdf.

[12] D. Karrenberg, *Anycast and BGP Stability*, Presented at RIPE 50, March 2005. http://www.ripe.net/ripe/meetings/ripe-50/presentations/ripe50-plenary-tue-anycast.pdf.

[13] P. Boothe and R. Bush, *DNS Anycast Stability – Some* Early *Results*, Presented at 2005 APNIC Open Policy Meeting, February 2005. https://archive.psg.com/050223.anycast-apnic.pdf.

[14] Change 1954, BIND 9.4.0 Release Notes, https://lists.isc.org/pipermail/bind-announce/2007-February/000495.html.

Appendix A

A.1 Compile-time Configuration Options

The following compilation options were used for NSD:

```
--prefix=/usr
--with-ssl
--sysconfdir=/etc
--localstatedir=/var
--with-facility=LOG_LOCAL6
--enable-root-server
```

The following compilation options were used for BIND:

```
--prefix=/
--bindir=/usr/bin
--sbindir=/usr/sbin
--sysconfdir=/etc
--localstatedir=/var
--libdir=/usr/lib
--mandir=/usr/share/man
--includedir=/usr/include/bind
--with-libtool
--with-openssl
```

A.2 Run-time Configuration Options

The following configuration file was used for NSD latency testing:

```
# ------------------------------------
#       nsd.conf
# ------------------------------------

server:
        ip-address: 10.3.0.16
        ip-address: fd80:a6f1:a2a5:1::10
        database: "/var/nsd/nsd.db"
        logfile: "/var/log/nsd.log"
        pidfile: "/var/run/nsd.pid"
        zonesdir: "/var/nsd"
        server-count: 4

key:
        name: oarc-axfr-key
        algorithm: hmac-md5
        secret: [ommitted]

zone:
        name: "."
        zonefile: "root.zone.signed"
```

The following configuration file was used for BIND latency testing:

```
// ------------------------------------
//      named.conf
// ------------------------------------

include "/etc/rndc.key";

options {
        directory "/var/named";
        pid-file "/var/run/bind/run/named.pid";
        allow-transfer { none; };
        notify no;
        listen-on { 10.3.0.16; };
        listen-on-v6 { fd80:a6f1:a2a5:1::10; };
        recursion no;
        dnssec-enable yes;
};

controls {
        inet 127.0.0.1 allow { localhost; } keys { "oarc-key"; };
};

logging {
        channel namelog {
                file "/var/log/named/named.log" versions 3 size 20m;
                print-time yes;
                print-severity yes;
                print-category yes;
                severity info;
        };
        category security { null; };
        category update-security { null; };
        category default { namelog; };
};

zone "." IN {
        type master;
        file "root.zone.signed";
};
```

The following configuration file was used for the BIND master for AXFR testing:

```
// ------------------------------------
//      named_master_axfr.conf
// ------------------------------------

include "/etc/rndc.key";

options {
        directory "/var/named";
        pid-file "/var/run/bind/run/named.pid";
        allow-transfer { none; };
        notify no;
        listen-on { 10.0.1.2; };
        transfer-source 10.0.1.2;
        notify-source 10.0.1.2;
        recursion no;
        dnssec-enable yes;
        provide-ixfr no;
};

controls {
        inet 127.0.0.1 allow { localhost; } keys { "oarc-key"; };
};

logging {
        channel namelog {
                file "/var/log/named/named.log" versions 3 size 20m;
                print-time yes;
                print-severity yes;
                print-category yes;
                severity info;
        };
        category default { namelog; };
};

key "oarc-axfr-key" {
        algorithm hmac-md5;
        secret [ommitted]
};

key "oarc-ddns-key" {
        algorithm hmac-md5;
        secret [ommitted]
};

zone "." IN {
        type master;
        file "root.zone";
        notify explicit;
        also-notify { 10.0.2.2; };
        allow-transfer { key "oarc-axfr-key"; };
        allow-update { key "oarc-ddns-key"; };
        max-transfer-time-out 1440;
        sig-validity-interval 7;
};
```

The following configuration file was used for the BIND slave for AXFR testing:

```
// ------------------------------------
//       named_slave_axfr.conf
// ------------------------------------

include "/etc/rndc.key";

options {
        directory "/var/named";
        pid-file "/var/run/bind/run/named.pid";
        allow-transfer { none; };
        notify no;
        listen-on { 10.0.2.2; };
        transfer-source 10.0.2.2;
        notify-source 10.0.2.2;
        recursion no;
        dnssec-enable yes;
};

controls {
        inet 127.0.0.1 allow { localhost; } keys { "oarc-key"; };
};

logging {
        channel namelog {
                file "/var/log/named/named.log" versions 3 size 20m;
                print-time yes;
                print-severity yes;
                print-category yes;
                severity info;
        };
        category default { namelog; };
};

key "oarc-axfr-key" {
        algorithm hmac-md5;
        secret [ommitted]
};

server 10.0.1.2 {
        keys { "oarc-axfr-key" };
        request-ixfr no;
};

zone "." IN {
        type slave;
        file "root.zone";
        masters { 10.0.1.2; };
        max-transfer-time-in 1440;
};
```

The following configuration file was used for the NSD master for AXFR and IXFR testing:

```
# -------------------------------------
#       nsd_master_axfr_ixfr.conf
# -------------------------------------

server:
        ip-address: 10.0.1.2
        ip-address: fec0:0:0:1::2
        database: "/var/nsd/nsd.db"
        logfile: "/var/log/nsd.log"
        pidfile: "/var/run/nsd.pid"
        zonesdir: "/var/nsd"
        server-count: 1

key:
        name: oarc-axfr-key
        algorithm: hmac-md5
        secret: [ommitted]

zone:
        name: "."
        zonefile: "root.zone"
        notify: 10.0.2.2 oarc-axfr-key
        provide-xfr: 10.0.0.0/8 oarc-axfr-key
```

The following configuration file was used for the NSD slave for AXFR and IXFR testing:

```
# -------------------------------------
#       nsd_slave_axfr_ixfr.conf
# -------------------------------------

server:
        ip-address: 10.0.2.2
        ip-address: fec0:0:0:2::2
        database: "/var/nsd/nsd.db"
        logfile: "/var/log/nsd.log"
        pidfile: "/var/run/nsd.pid"
        zonesdir: "/var/nsd"
        server-count: 1

key:
        name: oarc-axfr-key
        algorithm: hmac-md5
        secret: [ommitted]

zone:
        name: "."
        zonefile: "root.zone"
        allow-notify: 10.0.0.0/8 oarc-axfr-key
        request-xfr: 10.0.1.2 oarc-axfr-key
```

The following configuration file was used for the BIND master for IXFR testing:

```
// ------------------------------------
//      named_master_ixfr.conf
// ------------------------------------

include "/etc/rndc.key";

options {
        directory "/var/named";
        pid-file "/var/run/bind/run/named.pid";
        allow-transfer { none; };
        notify no;
        listen-on { 10.0.1.2; };
        transfer-source 10.0.1.2;
        notify-source 10.0.1.2;
        recursion no;
        dnssec-enable yes;
};

controls {
        inet 127.0.0.1 allow { localhost; } keys { "oarc-key"; };
};

logging {
        channel namelog {
                file "/var/log/named/named.log" versions 3 size 20m;
                print-time yes;
                print-severity yes;
                print-category yes;
                severity info;
        };
        category default { namelog; };
};

key "oarc-axfr-key" {
        algorithm hmac-md5;
        secret [ommitted]
};

key "oarc-ddns-key" {
        algorithm hmac-md5;
        secret [ommitted]
};

zone "." IN {
        type master;
        file "root.zone.1000000";
        notify explicit;
        also-notify { 10.0.2.2; };
        allow-transfer { key "oarc-axfr-key"; };
        allow-update { key "oarc-ddns-key"; };
        max-transfer-time-out 1440;
        sig-validity-interval 7;
```

The following configuration file was used for the BIND slave for IXFR testing:

```
// ------------------------------------
//      named_slave_ixfr.conf
// ------------------------------------

include "/etc/rndc.key";

options {
        directory "/var/named";
        pid-file "/var/run/bind/run/named.pid";
        allow-transfer { none; };
        notify no;
        listen-on { 10.0.2.2; };
        transfer-source 10.0.2.2;
        notify-source 10.0.2.2;
        recursion no;
        dnssec-enable yes;
};

controls {
        inet 127.0.0.1 allow { localhost; } keys { "oarc-key"; };
};

logging {
        channel namelog {
                file "/var/log/named/named.log" versions 3 size 20m;
                print-time yes;
                print-severity yes;
                print-category yes;
                severity info;
        };
        category default { namelog; };
};

key "oarc-axfr-key" {
        algorithm hmac-md5;
        secret [ommitted]
};

server 10.0.1.2 {
        keys { "oarc-axfr-key" };
};

zone "." IN {
        type slave;
        file "root.zone";
        masters { 10.0.1.2; };
        max-transfer-time-in 1440;
};
```