Wow, That's a Lot of Packets

Duane Wessels, Marina Fomenkov

Abstract—**Organizations operating Root DNS servers report loads exceeding 100 million queries per day. Given the design goals of the DNS, and what we know about today's Internet, this number is about two orders of magnitude more than we would expect.**

With the assistance of one root server operator, we took a 24-hour trace of queries arriving at one of the thirteen root servers. In this paper we analyze these data and use a simple model of the DNS to classify each query into one of nine categories. We find that, by far, most of the queries are repeats and that only a small percentage are legitimate.

We also characterize a few of the "root server abusers," that is, clients sending a particularly large number of queries to the root server. We believe that much of the root server abuse occurs because the querying agents never receive the replies, due either to packet filters, or to routing issues.

Keywords—**DNS root server**

I. Background: *DNS 101*

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

Each zone has one or more authoritative name servers. These are dedicated servers, whose job is to answer queries for names within their zone(s). For example, UCSD has three authoritative name servers. An application that needs to know the IP address for `www.ucsd.edu` can send a DNS query to one of those servers, which then returns an authoritative answer. If the application does not know where to send a query it asks the servers in the parent zone. In the example above, not knowing anything about `ucsd.edu`, the application should send a query to the authoritative server for the `edu` zone. If the application does not know about the `edu` zone, it queries the "root zone." This process is called *recursive iteration*.

The DNS root zone is served by 13 name servers (not to be confused with the 13 generic top-level domain servers) distributed across the globe. Thirteen is the maximum number of root servers possible in the current DNS architecture because that is the most that can fit inside a 512-byte UDP reply packet. Ten root servers are located in the U.S., two are in Europe, and one is in Asia.[1] The root zone and the root name servers are vital because they are the starting points for locating anything in the DNS. Without them, the DNS and hence almost every application we use (the Web, ssh, email) would be rendered unusable.

DNS clients, or resolvers, that query name servers, come in one of two flavors: stub and recursive. Stub resolvers, typically found in user applications, such as web browsers, ssh clients, and mail transfer agents, are rather primitive and mostly rely on smarter recursive resolvers that understand name server referrals. Recursive resolvers are usually implemented in specialized DNS applications such as the Berkeley Internet Domain Name (BIND) [2] server and Microsoft's DNS server. Most organizations operate local recursive name servers.

Recursive name servers cache name server responses, including referrals. Caching conserves network resources because intermediate servers do not need to query the root name servers for every request. For example, the name server learns that `a.gtld-servers.net` and others are authoritative for the `com` zone and sets the time-to-live (TTL) for this information. Typical TTLs for top level domains are on the order of 1–2 days.

In theory, a caching recursive name server only needs to query the root name servers for an unknown top level domain or when a TTL expires. However, a number of studies have shown that the root name servers receive many more queries than they should. In this paper we thoroughly investigate and characterize root name server traf-

The Measurement Factory, Inc., Boulder, Colorado, E-mail: `wessels@measurement-factory.com`.

CAIDA, San Diego Supercomputer Center, University of California, San Diego. E-mail: `marina@caida.org`.

Support for this work is provided by WIDE and DARPA NMS N66001-01-1-8909.

[1] In fact many of the root name servers are actually multiple hosts behind network load balancers. Some of them even occupy a few physical locations, employing IPv4 anycast to operate under a single IP address.

fic using 24 hours of *tcpdump* data collected at the F root server, `f.root-servers.net`. Our goal is to study the statistics of queries to a root server in order to understand how the valuable root server resources are used. We also consider the impact of non-caching referrals and attempt to identify applications and/or operating systems that are more likely to be root server abusers.

II. Related Work

Analysis of root server logs by Danzig *et al.* [3] was the first published large-scale study of DNS. They found a large number of implementation errors that caused unnecessarily wasted bandwidth. Not only are some of these bugs still with us today, ten years later, but the explosive Internet growth during the last decade has exacerbated this problem by orders of magnitude. Sadly, the authors' prediction of "buggy new implementations arising from vendors" also turned out to be true and "defective resolvers and name servers" continue to generate millions of meaningless wide-area packets every day.

Nemeth *et al.* [4] analyzed a few hours of traces collected at the F root name server and found an astounding number of bogus queries. Our study, based on a larger volume of data, continues and further extends their analysis. We compare our findings with theirs in Section IV-C.

Several projects are devoted to continuous monitoring of the DNS root servers' performance [5], [6], [7]. Their results are posted on the Web and are updated at least daily.

III. Methodology

A. Trace Collection

The F root server consists of four machines located in San Francisco and Palo Alto. We measured query traffic for all four machines for 24 hours on October 4, 2002. We captured only queries (but not responses) and stored them as raw *tcpdump* packets. The total trace file is 14 GB and contains 152,744,325 queries. Figure 1 shows that the rate of queries during our observations exhibited rather large but brief spikes. The mean query rate was 1768 per second. Note that the general pattern of query traffic remains the same throughout the 24-hour observation period.

Note that we did not actually run *tcpdump* on the root server itself. Rather, the server's switch is configured for port mirroring, and we run *tcpdump* on a separate host. Thus our monitoring activities are less likely to interfere with the root server's operation. Furthermore, `f.root-servers.net` utilizes packet rate limiting so that any single source does not overwhelm the server. Each source IP address is limited to 10 Kbits/sec, with a queue size of 3 packets. Packets that would exceed the queue size are simply dropped and never delivered to the server application. Thus our data represent the traffic arriving at the server host, but not the traffic presented to the name server software (BIND).

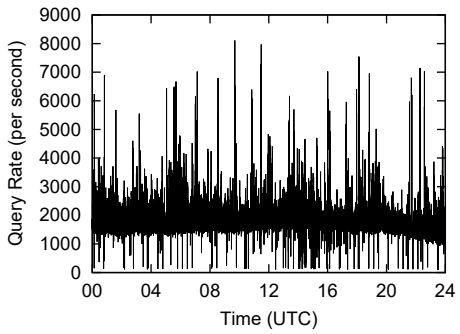

Fig. 1. Number of queries per second during 24-hour observation period on 4 October 2002 at the F-root DNS server.

During the time of our trace, `f.root-servers.net` was authoritative for more than just the root zone. In particular, it was authoritative for `edu`, `gov`, `arpa`, `in-addr.arpa`, and `root-servers.net` itself. This means that F receives more queries than it would were it only authoritative for the root zone. Due to the nature of DNS, we cannot eliminate queries based on the server's role. F may have received a query for `foobar.edu` because it is an `edu` name server, or because it is a root server.

The decision to capture only queries, and not responses, comes from our desire to take a minimal amount of data from the root server. The additional information may have allowed us to perform additional, or more sophisticated, analyses. However, adding responses to the trace would almost double the amount of data we dealt with.

B. Classifying Queries

We made the following assumptions about queries:

- Each source IP address is a single DNS agent. In our analysis of caching behavior we assume that a subsequent query from a given IP address comes from the same agent as the first query. In fact, it is possible that two separate agents share an IP address, but we ignore such cases.
- DNS agents are caching name servers. As we discuss in Section IV-A, some queries appear to come from stub resolvers. However, we believe that only caching, recursive name servers should be talking to the root servers. Therefore, we consider non-caching as an illegitimate behavior.
- DNS agents are not restarted, thus losing their caches, during the 24-hour period.
- The root server answers most queries, and these responses are received back at the source. Since our *tcpdump*

trace contains only queries, and not responses, we cannot be sure how many responses were actually sent.

We categorized each query in the trace into one of the following nine ordered categories. Note that each query is only placed into one category. For example, the trace may contain a repeated query with an unknown TLD, thus qualifying for two categories. We check for unknown TLDs before repeats, so such a query is classified only as the former. Our categories are:

B.1 Unused Query Class

The query class field in a DNS message occupies 16 bits and can have a value between 0 and 65535. However, the standards and implementations only define five values: IN (1), CHAOS (3), HS (4), NONE (254), and ANY (255). The CHAOS and HS (Hesiod) classes appear to be in support of MIT networking systems [8], [9]. The ANY class is a wildcard, and NONE is used for precondition checks in the DNS update protocol. No other classes are defined, and we would not expect to see queries with unknown classes arriving at a root name server.

B.2 A for A

This is, simply, an `A` query for a name that is already in numeric address form. For example:

```
06:45:38.573855 236.197.47.135.32772 > f.53:  25927 A? 207.244.8.2.
```

The application generating this query should be able to recognize that it already has an IP address and avoid the query altogether.

B.3 Unknown TLD

The Internet currently has 258 top-level domains (TLDs). These legitimate TLDs include country code domains (`fr`, `au`, etc.), the traditional generic domains (`com`, `net`, etc.), and some newer domains (`biz`, `info`, and others). A query for a name not matching one of the known TLDs is classified as an Unknown TLD.

B.4 Non-printable characters in query name

According to the DNS protocol specifications [10], the only valid characters in DNS names are the letters A–Z, numbers 0–9, and hyphen. This restriction is frustrating for people desiring to use native languages that have additional characters. A number of protocol modifications have been proposed ([11], [12]) and some applications are already using extended character sets. Nonetheless, we choose to separate out queries that contain characters outside the range defined by RFC 1035 [10].

B.5 RFC 1918 in PTR

RFC 1918 [13] defines network addresses for private, intranet use. Countless organizations use those subnets behind network address translation servers. In theory, these private addresses should never leak into to the public Internet. Section 3 of RFC 1918 states:

Indirect references to such addresses should be contained within the enterprise. Prominent examples of such references are DNS Resource Records and other information referring to internal private addresses. In particular, Internet service providers should take measures to prevent such leakage.

That is, outside observers (such as a root server) should not see IP packets with source or destination addresses that are in RFC 1918-specified address space. We also should not observe PTR queries for such addresses. DNS administrators must install reverse zone files for the RFC 1918 address space that they use, essentially pirating the DNS for this space, and then make sure there is no access to these zone files from the external global Internet.

B.6 Identical Queries

An identical query occurs when a source sends a query with exactly the same parameters (such as class, name, type, and ID) as any of its previous queries. In our analysis we did not place any temporal restrictions on detecting an identical query. The time between repetitions may be a few microseconds or a few hours.

B.7 Repeated Queries

This category is similar to an Identical Query, except that only the class, name, and type must be the same, but the query ID values are different. Again, we do not impose time requirements on repeated queries.

Detecting repeated queries and uncached referrals (below) requires maintaining a history of previous queries. Thus we sort the trace by source IP addresses and analyze one IP address at a time.

B.8 Referral Not Cached

A referral-not-cached query occurs when a client sends a query for a different name in the same zone as its previous query. For example, assume that a client makes a query for `foo.com`. The root name server replies with a referral to the authoritative name servers for the `com` zone. The client should then send all future queries for names in `com` domain to those other name servers. If the root server receives a query for `bar.com` from this client, we classify it as referral-not-cached.

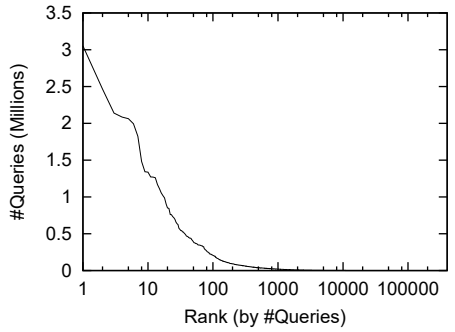

Fig. 2. Number of queries vs. rank.

QTYPE	Count	Percent
A?	84,710,847	55.5
PTR?	30,462,666	19.9
AAAA?	7,213,988	4.7
MX?	7,019,561	4.6
A6?	6,900,619	4.5
SOA?	6,403,621	4.2
ANY?	4,786,327	3.1
NS?	2,636,004	1.7
SRV?	1,819,762	1.2
CNAME?	662,553	0.4
other	128,377	<0.1

TABLE I
COUNT AND PERCENTAGE OF QUERY TYPES

Note that the F root server is authoritative not only for the root zone, but also for `arpa`, `in-addr.arpa`, `edu`, `gov`, and `root-server.net`. This complicates our analysis somewhat. Consider a query for `school1.edu` followed by `school2.edu`. We do not categorize the second query as referral-not-cached because F-root is the authoritative server in this case and the second query is legitimate. However, a query for `www.school1.edu` followed by `ftp.school1.edu` does get categorized as referral-not-cached.

B.9 Legitimate

Any query not matching one of the previous classes is deemed legitimate.

IV. RESULTS

A. General statistics

Our data contain 152,744,325 queries from 382,708 unique source IP addresses, or nearly 400 queries per source on average. However, the distribution of number of queries per source is extremely skewed. The 220 busiest sources generated 50% of the queries (nearly 350,000 queries per source on average). Figure 2 shows the source's rank (in the order of decreasing number of queries) on the x-axis and the number of queries per source on the y-axis. The #1 ranking source alone sent more than three million queries during the 24-hour observation period.

Table I presents the breakdown of queries by type. Not surprisingly, more than half are for `A` records. `PTR` queries come in second, comprising almost 20% of the total.

We found that 3,389,462 queries (2.22%) from 23,945 sources (6.26%) have the "recursion desired" (RD) bit set. The root name servers have recursion disabled, so many of these queries may be going unresolved. The RD bit is usually set by stub resolvers. The fact that stub resolvers are sending queries directly to root servers is a disconcerting discovery. It may indicate that incompetent system administrators are inserting the root server addresses into `/etc/resolv.conf` (or equivalent) files.

B. Busy Sources

We are particularly intrigued by those sources that generate the most queries. If the system were functioning properly, it seems that a single source should not need to send more than 1000 or so queries to a root name server in a 24-hour period. Yet we see millions of queries from certain sources in the trace.

We identified a few extraordinarily busy sources and analyzed the types of queries they generate, such as query names, types, IDs, and query interarrival times. Some of our findings follow. Source IP addresses are anonymized for privacy protection.

B.1 Source 1

The busiest source is from within a /8 network allocated to a branch of the U.S. military. This host generated 3,052,825 queries, or 2.00% of the total for our trace. Its average rate was 35 queries per second.

76.4% of the queries from this source are for a single name:

```
00:00:01.961516 160.30.209.71.1069 > f.53:  118 ANY? BURRBXR1.
00:00:01.961525 160.30.209.71.1069 > f.53:  8318 ANY? BURRBXR1.
00:00:01.961533 160.30.209.71.1069 > f.53:  6272 ANY? BURRBXR1.
00:00:01.961593 160.30.209.71.1069 > f.53:  8331 ANY? BURRBXR1.
00:00:03.027409 160.30.209.71.1069 > f.53:  10592 ANY? BURRBXR1.
```

The time between some queries is short—less than a millisecond in the above example. This is astonishingly shorter than the retransmission interval should be. The DNS protocol specification recommends that DNS clients wait 2–5 seconds before retransmitting a query.

Apart from being repeated so frequently, the other problem with these queries is that the name `BURRBXR1` is not

even a valid TLD. In the query classification (Table II below), these queries are counted as Unknown TLD.

Finally, this particular source address does not seem to be in the global routing table. A *traceroute* to this address stops at the first router without a default next-hop. The RouteViews server at the University of Oregon [14] reports "Network not in table."

B.2 Source 2

The second busiest source belongs to an organization providing Internet registry services. This source produced 2,465,092 queries. Of these, 58.5% were requests for the IP addresses of the 13 root name servers. Another 10% were for addresses within the organization's own zone. Interestingly, this source almost always sent queries in groups of three: for A, A6, and AAAA records. In other words, for each hostname it asked for an IPv4 address and for two types of IPv6 addresses.

The organization responsible for this source actually contributed 29 additional sources to our trace file. All together they sent 23,300,020 queries during the 24 hour period, or 15.3% of the total. Fortunately we were able to exchange email with someone at this organization. He explained that they use packet filters in an attempt to ensure that their servers provide answers for their authoritative zones only. Unfortunately they implemented inbound packet filters, but not outbound. Such configuration allowed their DNS servers to send queries to the root server but blocked the root server's responses.

Although we helped this company understand their misconfiguration, they did not feel compelled to fix the problem right away. Although they said it would be fixed with their "upcoming DNS migration," the high rate of query traffic remains as of early December 2002.

B.3 Source 3

The third busiest source is a customer of a DSL service provider. This host sent 2,138,697 queries to the root server. Almost all of these (99.96%) are `PTR` queries for IPv4 addresses in the `in-addr.arpa` zone. Furthermore, 88.6% of this client's queries are for a single name. Based on the answer to this frequently repeated query, it seems likely that the source is querying for the name of a host within its own organization.

We also noticed that queries from this particular source come from two different UDP source ports at the same time. This observation leads us to believe that there may actually be two name servers sharing this IP address.

This source also sent 109 `A` queries for this interesting name in the `in-addr.arpa` zone:

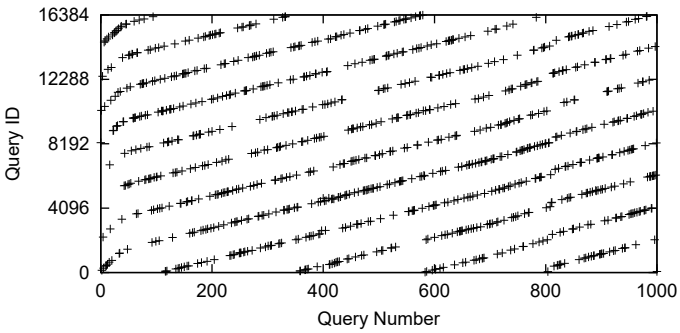

Fig. 3. Query ID vs Query Number for a client that only utilizes 25% of the Query ID range. (24 hours at F rootserver on 4 October 2002.)

```
A? 209.17.66.80.196.200.64.in-addr.arpa.
```

Names in the `in-addr.arpa` zone usually have four numeric components, but this one has seven. It seems like some buggy software must be generating these bogus queries.

B.4 Source 4

This source, actually the sixth busiest, is representative of a large group of sources that exhibit an interesting phenomenon. The valid range for the query ID field in a DNS message is 0–65535. However, many root server abusers send queries with ID values strictly less than 16384. Furthermore, they are neither sequential, nor random, but have the pattern shown in Figure 3. We are wondering what software generates DNS queries with this strange ID pattern.

It appears as though this DNS agent generates query IDs from a set of (eight) monotonically increasing counters. The counters do not increment sequentially, however. For some sources the pattern is not so obvious because the root server does not receive each query sent by the agent. However, the query IDs from these mystery agents also have another interesting property. Some ID values are more popular than others, and some values do not appear at all. Figure 4 shows a distribution of query ID values for Source #4. A quick glance shows that certain values (such as 0–7) are never used, and that the histogram has strong periodicity. Closer examination shows that the counts for non-occurring IDs are apparently shifted down. For example, the seven IDs from 65–71 never occur, but the ID 64 occurs eight times more frequently than the baseline. Similarly, IDs 49, 51, 53, and 55 never occur, but 48, 50, 52, and 54 occur about twice as often as the baseline. Perhaps this application is masking off certain query ID bits in some cases for some reason.

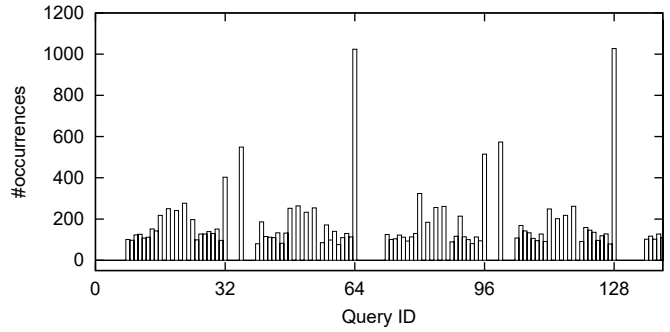

Fig. 4. Query ID vs Query Number for a client that only utilizes 25% of the Query ID range. This plot shows that some ID values are more common than others.

Type	Count	Percent
Unused Query Class	36,313	.024
A for A	10,739,857	7.03
Unknown TLD	19,165,840	12.5
Nonprintable in query	2,962,471	1.94
RFC1918 PTR	2,452,806	1.61
Identical Query	38,838,688	25.4
Repeated Query	68,610,091	44.9
Referral Not Cached	6,653,690	4.36
Legitimate	**3,284,569**	**2.15**

TABLE II
QUERY CLASSIFICATION RESULTS (24-HOUR PERIOD ON 4 OCTOBER 2002 AT THE F-ROOT DNS SERVER).

C. Query classification

Table II shows the count and percentage breakdowns for the nine query classifications defined in Section III-B. The categories are listed in the order of filtering applied to the data. Each query is categorized only once and an intersection among different categories is empty. For example, even if exactly the same query for an unknown TLD is repeated a million times, it is only counted in Unknown TLD category, but not in Identical Query.

Clearly, repeated queries, that are either exactly the same, or with different IDs but with the same content, represent the biggest category of DNS pollution at the root server. These two categories account for 70% of the observed query traffic!

Table III compares percentages of different types of queries in our measurements of the F root server query traffic with those by Nemeth *et al.* [4]. Note that this comparison is not straightforward since categories we used are mutually exclusive and categories in [4] are not. Therefore in their analysis each query may be counted more than once if it is illegitimate in multiple ways.

Type	Jan 2001	Oct 2002
A for A	12–18	7.03
Unknown TLD	20	12.5
RFC1918 PTR	7	1.61
Identical&Repeated Query	85	70.3

TABLE III
STATISTICAL COMPARISON OF OUR RESULTS WITH THOSE OF NEMETH ET AL. 2001 STUDY.

D. Well-Behaved Sources

We define a source as well-behaved if its number of legitimate queries is greater than the number of referrals not cached, repeated queries, and identical queries.[2] We chose these three classes because they probably indicate either buggy software used by a root server client, or configuration problems external to the client. Arguably we could also compare to the number of RFC1918 PTR queries, which are due to misconfigured name servers.

For example, the well-behaved source with the largest number of Legitimate queries (3222) also had 283 Referral Not Cached, 26 A-for-A, 107 Unknown TLDs, and 64 Repeated Queries.

By this definition the trace contains 310,608 well-behaved sources. In other words at least 72,100 (19%) of sources are not well-behaved.

E. TLDs and Query Names

Table IV shows the distribution of the most frequently requested unknown TLDs. This data provides some insight into how some computer systems may be misconfigured such that they generate substantial unnecessary DNS traffic. For each unknown TLD, the table shows its overall rank among all TLDs, the number of queries, the percentage relative to all queries, and the number of sources querying for that TLD.

The most popular unknown TLD comes from the single source described in Section IV-B.1. However, next in the list is “local,” which comes from 24,000 different sources. It seems likely that either (a) some relatively popular software is pre-configured with `local` as a domain name, or (b) many system administrators have chosen this domain for their private intranets and have failed to properly configure their internal name servers.

[2]Not greater than the sum of these three classes, but greater than each of them individually

TLD	Rank	Count	%	Sources
burrbxr1	6	2,331,857	1.53	1
local	7	2,001,210	1.31	24,123
localhost	9	1,962,413	1.28	12,428
wpad	16	651,230	.426	7107
test	32	225,961	.148	1185
eder003	33	218,210	.143	2
domain	41	162,682	.107	5083
lan	43	142,791	.093	2349
workgroup	47	121,547	.080	8471
5⟨C7⟩⟨D0⟩⟨B3⟩⟨E2⟩	50	110,880	.073	25
elvis	51	106,654	.070	49
admin	59	94,482	.062	821
ns1	68	71,556	.047	1519
msft	86	56,652	.037	1457
kornet	87	54,564	.036	2416
corp	92	51,776	.034	2115
loc	96	51,186	.034	1349
mailhost	97	50,698	.033	512
rcnet	109	48,196	.032	4
server	110	48,075	.031	5773
localdomain	119	46,462	.030	5255

TABLE IV

QUERIES FOR THE TOP 20 UNKNOWN TLDS. THESE STATISTICS COME FROM THE ENTIRE DATA SET, NOT JUST THOSE QUERIES CLASSIFIED AS UNKNOWN TLD. THIS LIST OMITS NUMERIC TLDS, WHICH ARE LIKELY DUE TO THE BOGUS A-FOR-A QUERIES.

"localhost" is, of course, a well-known name for a system's loopback interface. Many different applications use the name `localhost` when they need to communicate with the local system. In theory, network administrators should add an entry for `localhost` in their zone files. Obviously this doesn't always happen.[3]

The non-TLD "wpad" comes from the Web Proxy Auto Discovery protocol, put forth years ago by Microsoft and Inktomi. Web user agents are supposed to prepend `wpad` to their domain names and issue an `A` query. However, it appears that some implementations issue this query even if the system's domain name is not set.

Table V shows a list of the most popular query names. We see some names in common with the TLD list (`burrbxr1`, `localhost`, `wpad`). Many of the popular names are for the root servers themselves. The `in-addr.arpa` domain is also very popular, although queries for those names typically come from a small number of sources.

[3]In the course of investigating this, we discovered that *k.root-servers.net* was serving localhost as a valid TLD, although not during the time of our trace collection. It has since been removed.

Query Name	Count	Sources
.	2,982,245	50,031
burrbxr1	2,331,857	1
localhost	1,957,225	11,921
19.31.72.166.in-addr.arpa	1,894,433	15
b.root-servers.net	1,624,504	5808
d.root-servers.net	1,609,124	5828
i.root-servers.net	1,605,421	5708
a.root-servers.net	1,592,555	5923
f.root-servers.net	1,589,174	5406
c.root-servers.net	1,585,910	5759
h.root-servers.net	1,585,275	5826
e.root-servers.net	1,584,081	5788
g.root-servers.net	1,580,213	5781
l.root-servers.net	1,562,124	4726
m.root-servers.net	1,553,264	4457
k.root-servers.net	1,539,592	4770
j.root-servers.net	1,531,589	4773
philorch.com	1,186,696	1
⟨nonprintable⟩.tw	987,508	4
⟨nonprintable⟩.2ndpower.com	665,414	2
wpad	651,230	7107
auto.search.msn.com	638,282	1241
25.0/26.96.189.203.in-addr.arpa	559,341	2
www.opasoft.com	527,190	449
in-addr.arpa	503,076	15,089
mxmail.register.com	423,776	104
7.too.co.il	422,627	1
226.82.48.108.in-addr.arpa	372,789	2
136.154.213.54.in-addr.arpa	370,533	1
82.116.105.182.in-addr.arpa	367,112	27
www.math.uwaterloo.ca	365,227	7
tgp-gfn.trulyglobal.com	354,207	3
21.9.128.in-addr.arpa	321,712	1
154.85.72.18.in-addr.arpa	304,290	60
130.128/209.28.10.102.in-addr.arpa	295,150	136
104.193.151.229.in-addr.arpa	293,653	1
d11-c5.data-hotel.net	273,526	1

TABLE V

TOP 27 FULL QUERY NAMES SEEN IN THE 24 HOUR TRACE FILE. IP ADDRESSES HAVE BEEN CHANGED IN THE INTEREST OF PRIVACY.

F. OS Fingerprinting

We were curious to understand whether certain operating systems are more likely to be root-server abusers due to broken name server software. We used a simple OS fingerprinting trick to broadly categorize host operating systems: IP TTL values. Different operating systems use different initial values for the IP TTL field. BSD and Linux variants use 60 or 64, Microsoft Windows uses 128, and Solaris uses 255. By examining the TTL of received packets, we can infer their initial values, and hence their operating sys-

		Percent of Sources	
OS	TTLs	All	Busiest
BSD/Linux	35–64	49	17
Windows	100–128	40	58
Solaris/?	227–255	7.7	23

TABLE VI
BREAKDOWN OF OPERATING SYSTEM SENDING QUERIES TO THE F ROOT DNS SERVER ON 4 OCTOBER 2002, BASED ON IP TTL ANALYSIS.

tem. Table VI shows the result.

The table shows that most operating systems querying F root this day appear to be BSD or Linux based, with Windows a close second. The *Busiest* column shows the percentages for only the top 220 busiest sources (those contributing 50% of the total traffic).[4]

G. Repeat Interarrival Times

Because repeated and identical queries contribute to such a significant amount (70%) of the queries, we were interested in analyzing the time between repeats. A properly functioning name server should wait 2–5 seconds before retransmitting a query.

Recall that our categorization of queries is such that a particular query falls into only one category. In Section IV-B.1 we showed the busiest source sending the same query separated only by milliseconds. However, we classified those particular queries as Unknown TLD because that check appears first in our list.

Also recall that we have two categories of repeated queries: identical queries (with the same query ID) and repeated queries for the same name, but with different IDs. We analyze each category separately. Figure 5 shows the interarrival time distribution for identical queries. The distribution has a number of tall spikes between 48–92 seconds, with the tallest spike at 64 seconds. The spikes are separated by 4 seconds.

These appear to be due to an application that sends repeated queries for the root zone itself, with the same query ID, at fixed intervals:

```
1033694140.729078 159.47.2.99.32781 > f.53:  56716 NS? . (17) (DF)
1033694232.732738 159.47.2.99.32781 > f.53:  56716 NS? . (17) (DF)
1033694324.736630 159.47.2.99.32781 > f.53:  56716 NS? . (17) (DF)
1033694416.740175 159.47.2.99.32781 > f.53:  56716 NS? . (17) (DF)
1033694508.743738 159.47.2.99.32781 > f.53:  56716 NS? . (17) (DF)
1033694600.749451 159.47.2.99.32781 > f.53:  56716 NS? . (17) (DF)
1033694692.750852 159.47.2.99.32781 > f.53:  56716 NS? . (17) (DF)
1033694784.754556 159.47.2.99.32781 > f.53:  56716 NS? . (17) (DF)
```

Figure 6 shows the same information for Repeated Queries. However, here the distribution is quite different.

[4]Except that we removed the 30 abusers contributing 15% of total queries from the single organization as an anomaly.

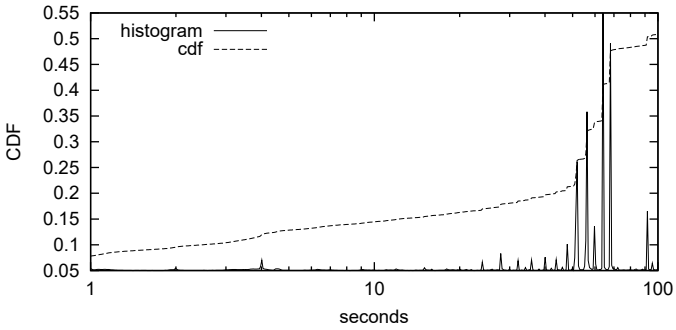

Fig. 5. Interarrival time histogram for Identical Queries

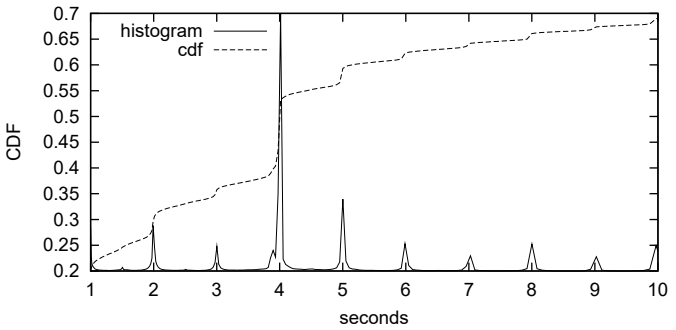

Fig. 6. Interarrival time histogram for Repeated Queries

The median, and tallest spike, is at four seconds. The data shows smaller spikes also at whole second intervals.

It appears as though most of the repeated queries are due to applications that correctly implement the retransmission suggestions from RFC 1035. We were initially surprised to find that most of the queries in this category, 45% of the total traffic, seem to be due to correctly functioning name server software. Why is it, then, that the F root name server receives more than 68 million repeated queries? Surely this amount cannot be explained by retransmissions due to packet loss.

We believe that the retransmissions occur because these sources never receive the root server's replies. We furthermore believe that this is largely due to packet filters at sources. For example, a product's default configuration may be to deny all packets until explicitly allowed by the administrator. In some cases, it may also be due to unidirectional routing problems. That is, reply packets are dropped because the source's IP address is unreachable or missing entirely from the routing table.

V. CONCLUSIONS

Packet filters on name servers are damaging. While they may protect your servers, they may also cause name servers to transmit significant amounts of useless traffic. If you must deny incoming packets, then also deny outgoing

packets as well.

We know some organizations use packet filters to make sure their name servers do not answer queries for non-authoritative domains. Instead, they should turn off recursion in the name server configuration.

Name server software should take more steps to detect and warn about possible misconfigurations. For example, sending hundreds or thousands of queries, without getting any answers probably indicates a low level communication problem. In this case, the name server application might rate limit itself, complain to syslog, or in extreme cases, exit altogether.

There should be a way for name servers to learn which TLDs are valid, and which are not. For example, if an SOA query for a TLD generates an NXDOMAIN reply, the name server could refuse to forward additional queries for that TLD for some amount of time.

Organizations throughout the Internet require education and enlightenment about how to properly use RFC 1918 address space. In particular, they must configure their local name servers to be authoritative for the appropriate `in-addr.arpa` zones.

REFERENCES

[1] P. Albitz and C. Liu, *DNS and BIND*, O'Reilly and Associates, 1998.

[2] "Bind website," `http://www.isc.org/products/BIND/`.

[3] P. Danzig, K. Obraczka, and A. Kumar, "An analysis of wide-area name server traffic: a study of the Internet Domain Name System," in *Proc. ACM SIGCOMM*, 1992.

[4] Evi Nemeth, k claffy, and Nevil Brownlee, "DNS Measurements at a Root Server," in *Proc. IEEE Globecom*, 2001.

[5] N. Brownlee, "Root/gTLD DNS performance plots," `http://www.caida.org/cgi-bin/dns_perf/main.pl`.

[6] K. Cho and et al., "A study on the performance of the root name servers," `http://mawi.wide.ad.jp/mawi/dnsprobe/`.

[7] R. Thomas, "DNS Data Page," `http://www.cymru.com/DNS/`.

[8] David A. Moon, "Chaosnet," Tech. Rep. A.I. Memo No. 628, 1981.

[9] Stephen P. Dyer, "The Hesiod Name Server," in *Proceedings of the USENIX Winter 1988 Technical Conference*, 1988.

[10] P. Mockapetris, "Domain names - concepts and facilities," Internet Standard 0013 (RFCs 1034, 1035), November 1987.

[11] M. Duerst, "Internationalization of Domain Names," December 1996, http://www.w3.org/International/1996/draft-duerst-dns-i18n-00.txt.

[12] Edmon Chung and David Leung, "DNSII Multilingual Domain Name Resolution," February 2001, http://www.ietf.org/proceedings/01mar/I-D/idn-dnsii-mdnp-02.txt.

[13] Y. Rekhter, B. Moskowitz, D. Karrenber, G. J. de Groot, and E. Lear, "RFC 1918: Address Allocation for Private Internets," February 1996.

[14] D. Meyer, "University of Oregon Route Views Project," `http://www.routeviews.org/`.