The Root of the Matter: Hints or Slaves

David Malone <dwmalone@cnri.dit.ie>

October 21, 2003

Abstract

We consider the possibility of having a name server act as a slave to the root zone, rather than caching information after it is requested. Tests, described here, indicate that this technique seems to be comparable to the traditional hints mechanism for moderately busy name servers and may offer other benefits such as reducing the number of bogus requests going to the root servers. With further refinement the technique may be operationally useful, but the impact on root servers would have to be assessed.

1 Introduction

The Domain Name System is a hierarchical distributed database that allows the lookup of records like `www.example.com`, where the dots separate the levels in the hierarchy. To look up a record you first ask a *root* name server for information about `www.example.com`, and it will respond with a message pointing you at the name servers for `.com`, then you send a message to one of these name servers and they will point you at the name servers for `.example.com` and finally you ask one of these, who provide you with information about `www.example.com`. Caching is well provided for within DNS and you are told how long to remember who the `.com` and `.example.com` are in the responses, reducing the load on the name servers closer to the root of the tree. Negative caching, ie. caching of records that are found not to exist, is also provided for. Typically negative responses are cached for a short time.

DNS requires very little boot-strap information and should be able to resolve any query by beginning with only a list of root name servers. Traditionally, DNS servers have been provided with a list of addresses that are root name servers at the time the DNS server software was written. The DNS server then queries these addresses requesting the current list of root name servers, and once it gets a response it is ready for operation.

To allow for redundancy within DNS, multiple name servers are permitted (and encouraged) for each point in the hierarchy. For ease of management, the DNS protocol defines a mechanism to transfer all the data at a point in the hierarchy from one name server to another. This mechanism is called a *zone transfer*. This means that the name servers for a zone may be either masters, who have first hand access to the data in the zone, or slaves, who copy it from a master using a zone transfer. A serial number and various lifetimes are included in the zone so a slave knows how frequently the master must be contacted to keep the zone up to date.

Some of the root name servers allow zone transfers of the root zone, meaning it is possible to to configure any name server as a slave for the root zone. Why would one want to do this? A server that has copy of a zone can respond to requests with a positive or negative answer immediately. In particular, this makes is possible to 'cache' negative results until the zone expires and a new copy of the zone must be transfered. However, it also means that you must transfer the complete zone even if you will only use a small proportion of it.

Our aim is to compare the traditional hints method with the alternative of acting as a slave for the root zone. There are two groups who might benefit from such a change. The first group is the users of the name server who may see a reduction in outgoing traffic and faster response times. The second group is the root name servers themselves, who might be receive less queries. Evidence already suggests that much of the traffic to the root name servers is in some way bogus and that if the bogus queries can be stopped at the local name server, that may be useful.

2 Setup

Our method is to monitor the traffic from two name servers to the root name servers over the period of two week. At the end of the week we change the name servers from the traditional hints method to be slaves for the root zone.

The two DNS servers were both running FreeBSD 4 and the name server software was BIND 9.2.1.

The first DNS server is used by a small research group of 10–20 people. It serves a quiet mail and web server and the workstations of the research group. We refer to this server as the *quiet server.*

The other DNS server is busy, acting as a name server to an academic department providing Unix services to 1000 undergraduates, postgraduates and staff, including a mail server, a web cache, a web server and an anonymous ftp server. These services will generate quite a high DNS load because the web server and TCP wrappers have been configured to do forward and reverse lookups for most connections. The name server also acts as a primary or secondary server for around 70 zones in each of two 'views'. This network also has IPv6 connectivity, the mail and web server are IPv6 capable, though the web cache is not. We refer to this server as the *busy server.*

The activity in the systems should be relatively independent — they are in different Universities, but are connected to the same ISP. The quiet system acts as a tertiary mail exchanger for the busy system.

```
#!/bin/sh
tcpdump -p -i fxp1 -n \
    -w SERVER-`date +%Y%m%d%H%M%S` \
    'host SERVER_IP and \
    port 53 and \
    ( host 198.41.0.4 \
    or host 128.9.0.107 \
    or host 192.33.4.12 \
    or host 128.8.10.90 \
    or host 192.203.230.10 \
    or host 192.5.5.241 \
    or host 192.112.36.4 \
    or host 128.63.2.53 \
    or host 192.36.148.17 \
    or host 192.58.128.30 \
    or host 193.0.14.129 \
    or host 198.32.64.12 \
    or host 202.12.27.33 )'
```

Figure 1: Tcpdump parameters used for recording DNS traffic

```
zone "." {
        type hint;
        file "root.cache";
};
```

Figure 2: BIND 9 config for hinting root zone

DNS traffic from both systems was collected using tcpdump starting on Tuesday 28 January 2003 and Tuesday 2 February 2003. Traffic from the quiet system was collected by running tcpdump on that system. Traffic from the busy system was collected by running tcpdump on a router that sees all external traffic. The tcpdump command line is shown in Figure 1. The name server on both hosts was restarted as measurements began on both Tuesdays to ensure the nameserver's cache was starting empty.

For the first week, both servers used the traditional hints mechanism configured as shown in Figure 2. For the second they both acted as slaves for the root zone and used the configuration shown in Figure 3, suggested by Doug Barton on the FreeBSD-stable mailing list.

Note that the busy name server op-

```
zone "." {
        type slave;
        file "s/root";
        masters {
                128.9.0.107;
                192.33.4.12;
                192.5.5.241;
        };
        notify no;
};
```

Figure 3: BIND 9 config for slaving root zone

erates two BIND views and so the root zone must be configured in both views. For the purposes of this experiment it was configured to obtain two independent zone transfers from the root name servers for the two views and place them in two different files. In operation it would only be necessary to get one zone transfer from the root name servers and then the zone could be transfered to other views locally.

At the time of writing, the root zone is about 55KB on-disk.

3 Results and Analysis

Figure 4 and Figure 5 show a summary of the traffic volume between the root name servers and our quiet and busy DNS servers respectively. Each graph shows the 1 hour means for the byte or packet rate for both the hint and slave schemes over the duration of the experiment. Calculating means over a shorter time scale naturally produces noisier, peakier results, but the general pattern is similar.

Note that the graphs only show the first 144 hours of the data; around hour 156 a network outage disconnected the quiet server from the Internet at large. This resulted in a huge peak in outgoing DNS traffic as BIND repeatedly reissued requests until the network was reconnected.

Several interesting observations come from examining the graphs. First, the traffic for both configurations is quite similar in volume — it is not obvious from the graph which method caused a smaller amount of traffic. Second, the slave configuration shows peaks, that correspond clearly to zone transfers for the quiet server. For the busy server there are additional peaks that are not attributable to zone transfers. We might expect no traffic to be sent to the root name servers between zone transfers, however we do see some chatter between them.

Table 1 shows some summary statistics for the experiment. We show the packet and byte rates for all traffic, traffic coming into our DNS server and traffic emanating out from our DNS server. For the slave cases we also show the ratio of the slave method to the corresponding rate in the hint method. The table indicates that acting as a slave decreases the number of packets and bytes tranmitted in both the busy and quiet cases. For the busy server it also results in a reduction in the number of packets received.

To avoid the spike in the traffic mentioned above, the summary statistics were calculated again over the first 6 days of the trace. The results are shown in Table 2. We can see that the statistics do not change significantly.

Further analysis of the (truncated) trace was performed by parsing the output of tcpdump. About 1% of the packets were truncated to an extent that it was not possible to process them further.

TCP based queries are not parsed well by tcpdump, because it does not do TCP stream reassembly, but all the TCP queries are zone transfers. Over the 144 hours of the truncated trace the quiet server performed 13 zone transfers and the busy server 26 zone transfers. The difference is explained by the fact that the busy server was performing a zone transfer for each view. As noted above, this is not necessary in a more refined configuration. The quiet server averaged 2.5s transferring 81kB in 111 packets per zone transfer. The busy server averaged 1.7s transferring 103KB in 134 packets. Note, that the root zone did not actually change 13 times during the experiment but its serial number is regularly incre-

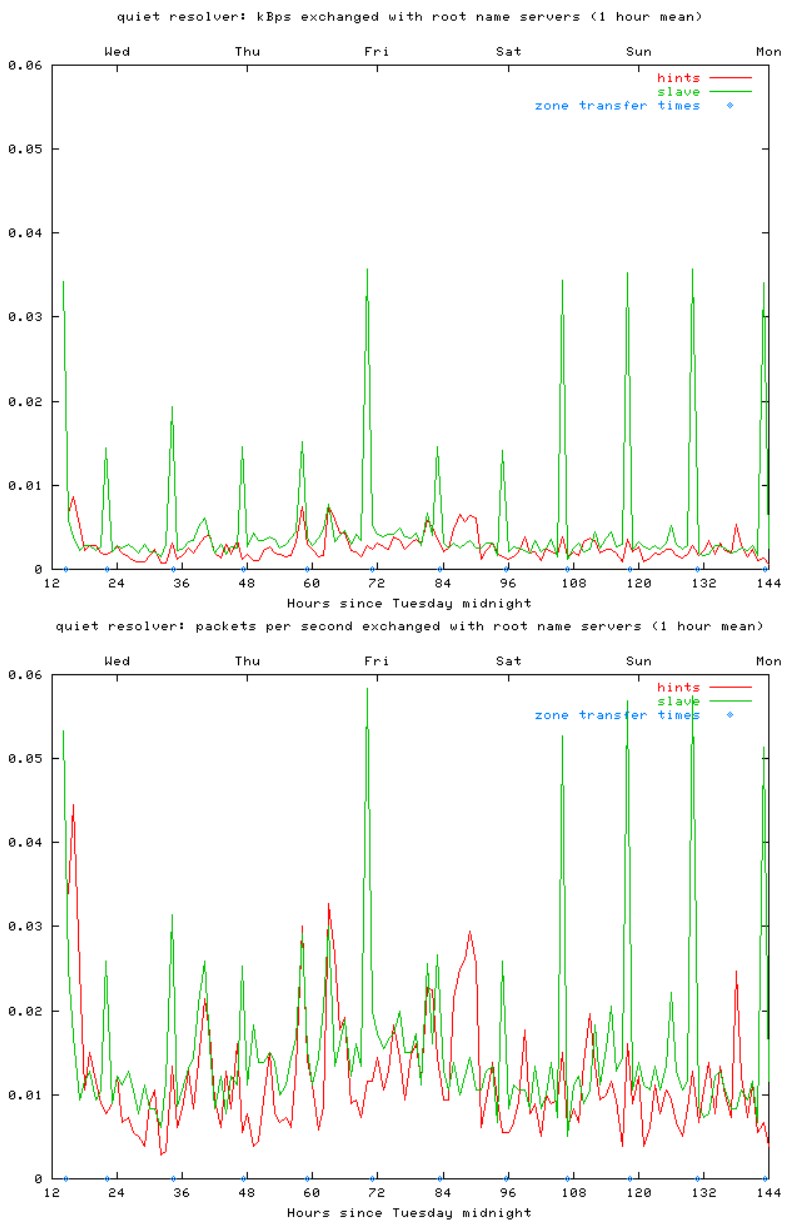
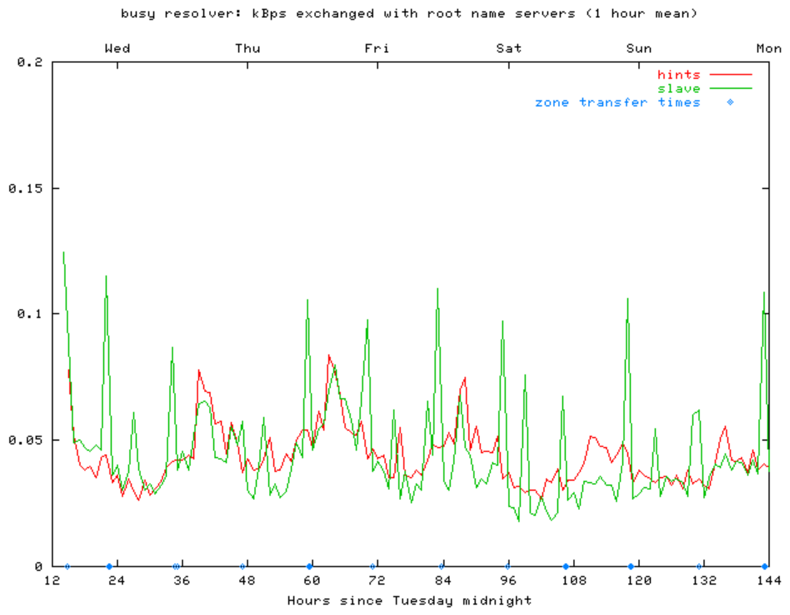


Figure 4: Quiet Server DNS Traffic

Direction	Packets/s	bytes/s	Packets/s	Bytes/s	Packets	Bytes
	quiet	**hints**	**quiet**	**slave**	**slave/hint**	
both	0.0215	3.2420	0.0156	5.1331	73%	158%
in	0.0059	2.1961	0.0079	4.6279	133%	211%
out	0.0157	1.0459	0.0078	0.5051	50%	48%
	busy	**hints**	**busy**	**slave**	**slave/hint**	
both	0.2151	46.8377	0.1753	47.4230	81%	101%
in	0.1075	39.2026	0.0878	41.0068	82%	105%
out	0.1076	7.6351	0.0875	6.4162	81%	84%

Table 1: Trace statistics, full trace



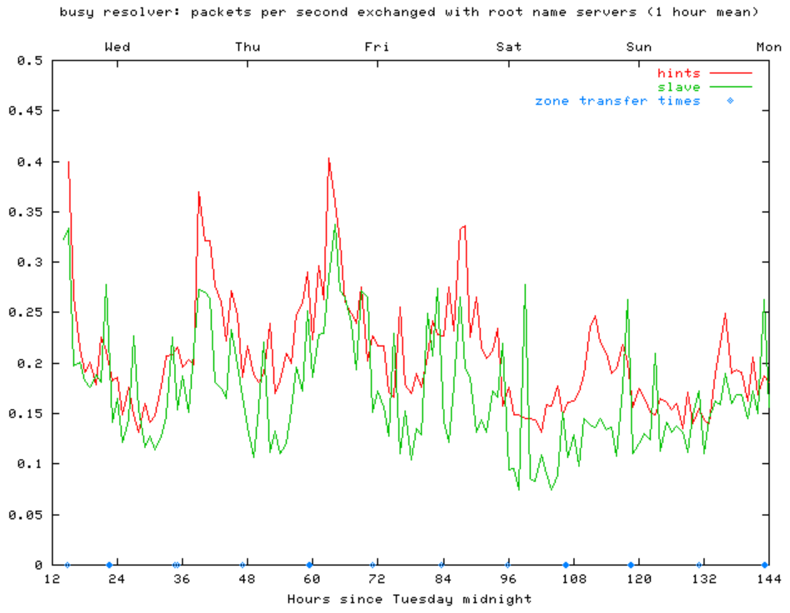


Figure 5: Busy Server DNS Traffic

Direction	Packets/s	bytes/s	Packets/s	Bytes/s	Packets	Bytes
	quiet	**hints**	**quiet**	**slave**	**slave/hint**	
both	0.0233	3.4348	0.0155	5.2111	67%	152%
in	0.0060	2.2772	0.0078	4.7125	130%	207%
out	0.0173	1.1576	0.0077	0.4986	45%	43%
	busy	**hints**	**busy**	**slave**	**slave/hint**	
both	0.2089	45.1614	0.1711	46.7172	82%	103%
in	0.1044	37.8130	0.0857	40.4514	82%	106%
out	0.1045	7.3484	0.0854	6.2658	82%	85%

Table 2: Trace statistics, truncated trace

mented to check that zone transfers to the public root servers are working correctly.

A breakdown by type of the queries is shown in Table 3. Note that in the table we have counted certain classes of bogus queries separately. In particular, requests for records corresponding to dotted quads or addresses enclosed in '[]' have been counted separately. This highlights a bug in a locally maintained MTA running near the busy server, as it is making requests for MX records for the latter.

These tables show that the number of queries made is significantly reduced in the slave case, as we expect. It also largely eliminates the obviously bogus requests that we have counted separately — this makes sense as the slave zone is a complete cache enabling the server to immediately determine that the top level zones 1–255 and 1]–255] do not exist.

The other startling thing highlighted by the table is the number of A6 queries being made by BIND 9.2.1. BIND's v6-synthesis option is not enabled on either server and to the best of our knowledge no resolver served by either server uses A6 records. The table also shows the SOA queries we expect BIND to make to ensure that its copy of the zone is up-to-date.

The table also shows the number of responses that resulted in an error response. All errors were actually of type NXDomain. We can see from the tables that when acting as a slave to the root zone, almost all NXDomain error are eliminated, as expected. In fact, in the busy server case, the reduction in NXDomain responses almost accounts for all but 785 queries saved by acting as a slave to the root zone.

Browsing the list of domains provoking NXDomain responses in the non-slave case shows domain names that are not fully qualified, domain names with typos (some obviously containing HTML, suggesting they have come from links in web pages, others seem likely to be URLs mistyped into browsers), Unix tty names and private names such as `localhost.localdomain` or `loopback`.

In all cases, a significant number of the NXDomain errors were associated with queries containing an additional resource record included in the query (denoted by [1au] in the tcpdump output). It seems these queries may be related to probing for DNSsec support. These accounted for 220 of the quiet-hint NXDomain, all 12 of quiet-slave NXDomain, 6531 of the busy-hint NXDomain and all 22 of the busy-slave NXDomain.

Finally, an attempt was made to determine how long BIND spent waiting for a response to queries. BIND will usually focus queries on remote servers with a low response time and probe other servers periodically to update estimates of response time. However, it is difficult to tell which queries are probes and which queries are real. Thus a wait time was calculated by summing min(response time, 5s) over all UDP queries (zone transfers are discussed above). Queries outstanding at the end of the trace are assumed to be answered at the time the trace ended.

The results are shown in Table 4. The quiet server shows a longer wait time when slaving the root zone and the busy server shows a shorter wait time.

4 Conclusion

The results seem to indicate that acting as a slave to the root zone results in a saving in the number of packets and queries transfered between the root zone. However, it is not as clear that it results in a reduction in the byte volume of traffic seems to be increased. It also involves a number of TCP connections to the root servers. As TCP connections are persistent, this will cause extra overhead on the root servers and it is unclear is the reduction in the number of queries would offset this load. Also TCP connections are less suitable for any-cast, though the short transfer times combined with the fact the zone does not immediately become invalid means this may not be a serious problem. If this technique went into wide use, the number of TCP connections could be significantly reduced

Type	Queries	Queries		NXDomain	NXDomain
	quiet hints	**quiet slave**	**slave/hint**	**quiet hints**	**quiet slave**
A	810	318		53	
A....	28			28	
A6	7469	1647		25	
A6....	28			28	
AAAA	147			73	
AAAA....	9			9	
AXFR		13			
MX	6				
PTR	380	343			12
SOA		970			
SRV	6			6	
total	8883	3291	37%	222	12
	busy hints	**busy slave**	**slave/hint**	**quiet hints**	**quiet slave**
A	11206	9158		1405	1
A....	315			314	
A6	31460	29418		33	
A6....	268			265	
AAAA	720	157		555	
AAAA....	53			53	
ANY	2			2	
AXFR		26			
MX	558			492	12
MX....	115	13		115	
MX[]	7899			7895	
NS	5				
PTR	660	647		7	9
SOA		1943			
SRV	24			24	
total	53285	41362	77%	11160	22

Types ending with '....' are queries for IPv4 dotted quads. Queries ending with '[]' are queries for addresses enclosed in brackets.

Table 3: Breakdown of query types and query errors

Wait time in seconds	
quiet hints	**quiet slave**
95.5	203.4
busy hints	**busy slave**
3700.3	2043.1

Table 4: Wait times

by only changing the root zone's serial number when the zone changed, or by incrementing the serial number less frequently for maintenence purposes.

We have not considered here the possibility of taking a single transfer of the root zone and then making your DNS a master server root zone, using that static data. This technique offers many of the benefits of the slaving technique described here; changes to the root zone are very infrequent and so the data is unlikely to become stale quickly. However when the root zone does change, this technique requires manual intervention.

One incidental benefit of using TCP based DNS is that TCP responses are not limited in size as UDP responses are and might simplify the addition of AAAA glue to the root zone.

One area where slaving the root zone seems particularly strong is in the reduction of bogus queries (dotted quads, [address], typos, unqualified domains).

A large number of queries made seem to be for A6 records. It is unclear why BIND is making these queries, but it may be attempting to obtain glue records for some reason. The responses to these queries will often obtain additional information records, meaning that the request may not be wasted, but it seems strange for BIND to direct these queries at the root servers at all. If these requests can are unnecessary and can be eliminated, then the gains of slaving the root zone would be larger.

Other minor issues identified include BIND's persistence when it does not receive a response from a root server, broken MX lookup code in a locally maintained MTA and the fact tcpdump should be run with the `-s 0` flag if circumstances allow.

5 Acknowledgements

I would like to thank Doug Barton and Niall Murphy for advice and feedback; João Damas and Jim Reid for answering my questions.

This work has been supported by Science Foundation Ireland under the National Development Plan.