A Measurement-Based Study of Optimal DNS Timeouts

Paul Kroon
Department of Mathematics and Computer Science
Ursinus College

***Abstract*— One of the first steps in communication between two Internet end systems is often a DNS lookup, mapping the domain name of the remote host to its IP address. This step introduces a compulsory delay that may have a very noticable effect on the overall loading of a Web page — particularly when a DNS query or response is lost. Depending upon server implementation, DNS timeouts may be quite high — five seconds or more — but the selection of timeout does not appear to be based on data taken from in-depth studies.**

In this paper we present a detailed investigation of the optimal timeout for DNS queries by DNS servers. To do so, we develop a method to investigate DNS response times, collect an appropriate sample of data, and propose an optimal timeout to be used by DNS server implementations. We found that the default timeout can be reduced to one second, noticably increasing user-perceived application performance while creating an imperceptible increase in network traffic.

I. Introduction

The Domain Name System, or DNS, was designed and implemented over twenty years ago and continues to perform exceptionally well despite the explosive growth of the Internet. DNS is primarily used to perform domain name lookups, or the mapping of domain names to IP addresses. Every time an Internet-based application makes a connection using a domain name, the DNS system is used to translate the name into an IP address useable by the machine. However, evidence exists to suggest that DNS delays can have a noticeable effect on user-perceived web performance [1], [2], [3], [4], [5], [6]. For example, Lee, *et al.* [3] found that the placement of root servers had a significant effect on DNS performance. Others, such as Cohen and Kaplan [1] and Wills and Shang [6] found that caching has a great impact on DNS performance and methods of increasing its functionality would have a positive effect on latency.

DNS works by sending queries to potentially multiple DNS servers, each responding with either the IP address being looked up or with the location of another server to query. This process is illustrated in Figure 1. A host first sends a request to its local DNS server (1). The local DNS server then sends queries on behalf of the host, following referrals given by the responses until it receives an authoritative response for the domain name (2-7). It can now respond to the querying host with this answer, allowing the user's program to accomplish the task it was performing (8).

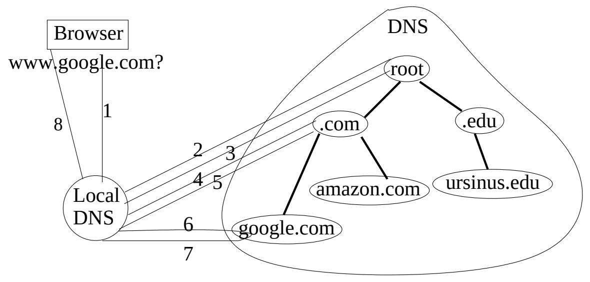

Fig. 1. The workings of DNS.

Originally, a minimum of two to five seconds was suggested for the timeout of queries [7]. Later, RFC1536 [8] stated that a good implementation of the DNS system was shown by BIND. It also stated that it starts ”at the greater of 4 seconds and 5 seconds divided by the number of servers” and then it ”cycles through servers and at the end of a cycle, backs off the time out exponentially,” continuing to 45 seconds. Since then, the various implementations of this system have either used this time as a default, or developed a new scheme of timeout related to this. However, no known in-depth study has an optimal time to use as the timeout. This paper addresses this omission.

The importance of an optimal timeout is its impact on the performance of web-based applications that make use of domain names. If the timeout for a query is too high, packet loss will result in a delay between the time that a packet is lost and the timeout. It is possible that this could be several seconds of latency because retransmission does not occur quickly enough. However, if the timeout is too low, then queries will be sent while a response to the original query is still being returned. This causes unnecessary redundant packets on the network, possibly negatively affecting performance, both of DNS and other network-based applications. The optimal timeout is the lowest possible time that keeps the number of redundant packets to an acceptable level.

For our research, we collected round-trip times for DNS queries. Using this data we analyzed the distribution of round trip times among DNS servers. This paper will discuss the methods we developed to collect our data and the effectiveness of a modified timeout and a proposed timeout if a different value is found to have a positive effect.

II. Related Work

A number of papers have examined performance of DNS and its relationship to performance of web-based applications. Mao, *et al.* [5] developed a method to measure the proximity between clients and their local DNS servers. They found that the configurations of local DNS servers on clients could be changed to decrease the distance between the two. In addition, they found that this would enhance the performance of the DNS system by measuring round-trip times of different geographical distances between clients and servers.

Also, Danzig, *et al.* [2] analyzed nameserver caching. The timeout methods of the BIND implementation of nameservers is mentioned to explain the way that it works and that it affects the performance of DNS.

Wills and Shang [6] found that about 20% of DNS lookups tested took more than one second, demonstrating a significant relationship between the DNS lookup time and the user-perceived time to load a web object.

Bent and Voelker [9] analyzed the user-perceived time to get an entire web page. They used a proxy that could duplicate the actions of a typically user in browsing various web pages, and collected the data related to this. Among their findings, the concluded that the average time for a DNS lookup for a single object was 7.1ms. They also found that the average DNS cost per page, at an average of 15 objects per page, was 529.7ms. We will later show findings that appear similar to these numbers.

Wessels, *et al.* [10] conducted laboratory simulations of the traffic on the root servers and TLDs. They examined the effects of various caching software on the name servers and how the different implementations spread the load on the servers. Among the data collected, they suggest that caching resolvers all follow the example of BIND 9 in its implementation and also should all have some time of exponential back-off.

Sekiya, *et al.* [11] developed a tool to collect the round-trip times to various name servers. They tested this tool on the root servers and several TLDs. Although their data is limited by the fact that they have samples from only 27 sites and only use dial-up connections in more remote areas, their findings also show that the RTTs all tend to be much lower than five seconds.

Brandhorst and Pras [12] collected traffic data at four different locations in order to analyze DNS. With the data they collected, they performed some statistical analysis to answer such questions as what the percentage of DNS traffic is, what some typical latencies are, and how often different types of responses are received.

Brownlee, Claffy, and Nemeth [13] also collected a large amount of data, but from a root server. With this data, they analyzed the queries and observed bogus queries, a few small DoS attacks, and examples of misconfigurations and their effects.

III. Methodology

To examine DNS timeouts, we began by investigating the distribution of round-trip times between any two nameservers. Using 2 million as an estimate of the number of nameservers [14], we calculated the number of different pairs of nameservers to be $\frac{2{,}000{,}000*(2{,}000{,}000-1)}{2} = 1.999999 * 10^{12}$. Because this was far too large a number of data points to collect, we determined that a sample of 1000 pairs of servers would provide a sufficient level of accuracy. To do the sampling we needed a list of nameservers. Because such a list does not exist , we developed a method to generate one. First, we gathered a list of domain names, as explained in section A. Then, we extracted the names and IP addresses of nameservers from these domain names, as described in section B. From this list, we gathered our data using the method explained in section C.

A. Gathering Domain Names

We began by gathering a large set of domain names. These names could be used to find the IP addresses of nameservers. To gather the domain names, we used a method similar to that of Gurevich and Bar-Yossef [15]. We automated Google queries using random combinations of two words from a wordlist containing roughly 250,000 words. The domain names in the top 100 results of each query were compiled into a list of 1,257,789 names.

B. Building List of Nameservers

Next we used the list of domain names to compile the IP addresses of nameservers. We created a script to send DNS queries and parse the responses. The script performed a *dig* query for an NS record on each of the names, and recorded the name and IP address of each nameserver it found. If it encountered an SOA or a CNAME record, it moved to the parent domain and added it to the bottom of the list to be queried again. For example, if *www.example.com* returned an SOA record, then *example.com* would be added to the bottom of the list to be tried again.

After completing this script, we had a file containing a list of name servers, each of which was listed with its IP address and a domain name for which it is authoritative. From this list, we randomly chose 2000 servers to create our final list of 1000 pairs. We made a program to randomly choose servers that were put into pairs. Every time a server was chosen, it was removed from the original list of name servers to ensure that a server was never used twice. At this point, we now had a list of 1000 pairs of servers with their domain name and IP address that we could use to collect RTT data.

C. Collecting Data

To gather the round-trip time data, we developed a three-step process , each of which used DNS queries and two nameservers. The times collected were the round-trip times between the two nameservers. Our method was to measure the time between our collection machine and the first nameserver (NS1) and the time between our machine and the second nameserver (NS2) through the first nameserver. This allowed us to calculate the time between the two nameservers. However, there were some complications to this method. One issue we encountered was the effect of caching, in both the nameservers and the routers between them. We had to take into account nameserver caching because we had to make sure that our queries went to the second nameserver and not just to the first. Also, the routers along the path can cache the route, which can also affect our measurements. The solution to this was to ensure that the path to NS1 was cached in the routers, and the IP address of NS2 and a domain it was authoritative for cached in NS1. This meant packets were subject to non-cached results only between the two nameservers as desired.

The first step was to send a query directly to NS1 to look up the IP address of the name of NS2. The purpose of this query was to cache the IP address of NS2 in the first nameserver without sending a query to NS2. Then, a second query was sent directly to NS1 for an NS record of the domain name for which NS2 is authoritative. This cached the domain name and the IP address of a nameserver that is authoritative for it in NS1, which we believed meant that further queries to NS1 for the same domain would go directly to NS2. Some possible problems with this are explained in section *V*.

The second step was to send a query directly to NS1 for the IP address of its own name. This is how we collected the time between our collection machine and NS1. This was repeated three times to provide higher quality data. The time for this query was recorded and is used to subtract from the total round-trip time to obtain only the time between NS1 and NS2.

In the final step a query was sent directly to NS1 asking for a non-existent domain name for which NS2 would be an authoritative server. This was

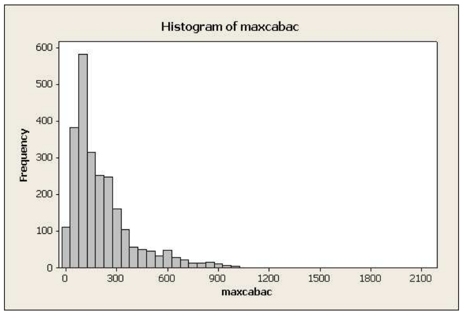

Fig. 2. Histogram of maxcabac.

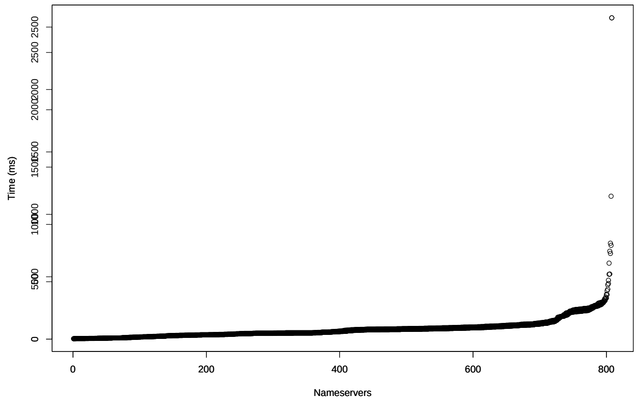

Fig. 3. Time for first query to NS1.

repeated three times and the time for the lookup to complete was recorded.

This process was repeated for 1,000 pairs of nameservers, a sample of all the servers collected in the list.

IV. Results

The results are shown in Figures 3 through 8.

From the similarities in the first three graphs, and the fact that over 99% of the data is under 500ms for all three of these graphs, it can be inferred that the data shown is consistant as appears to be valid. The outliers that can be found in each graph can be explained by congestion in either the route to the server or on the server itself. Further testing of the servers that produced these times showed that a more consistent time could be found, helping prove the congestion explanation.

In the other three graphs, the values for the rount-trip times for the query to the second nameserver (NS2) from the first are shown. The same trend is seen in these graphs. The majority of values are on

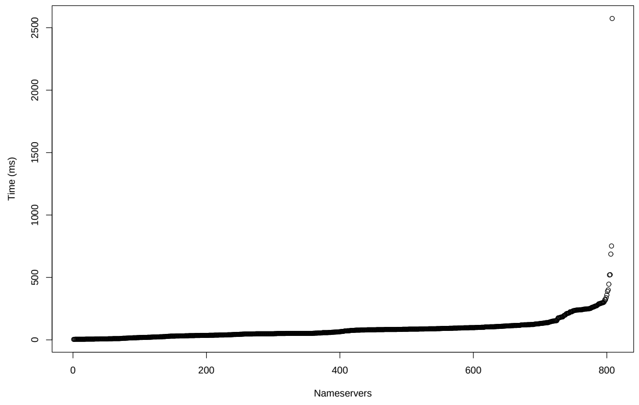

Fig. 4. Time for second query to NS1.

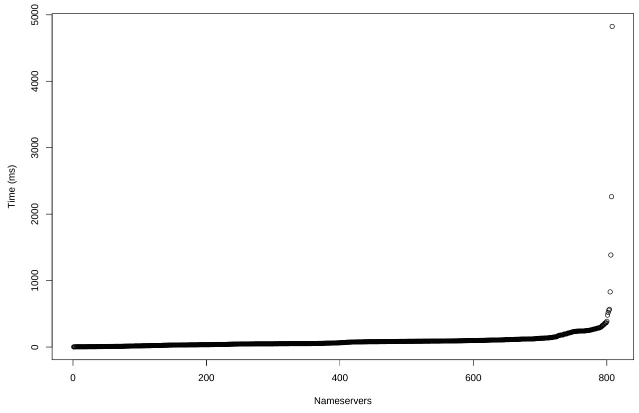

Fig. 5. Time for third query to NS1.

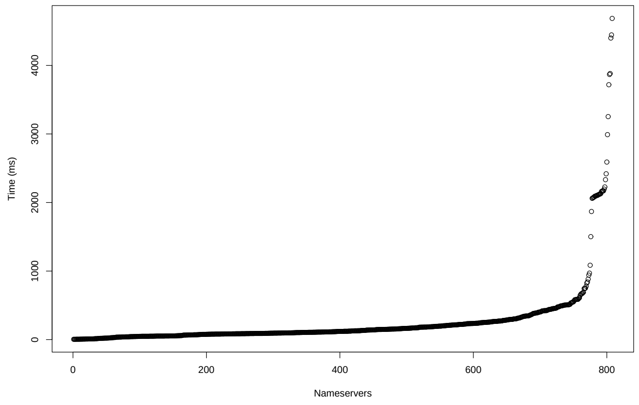

Fig. 6. Time for first query to NS1, asking NS2.

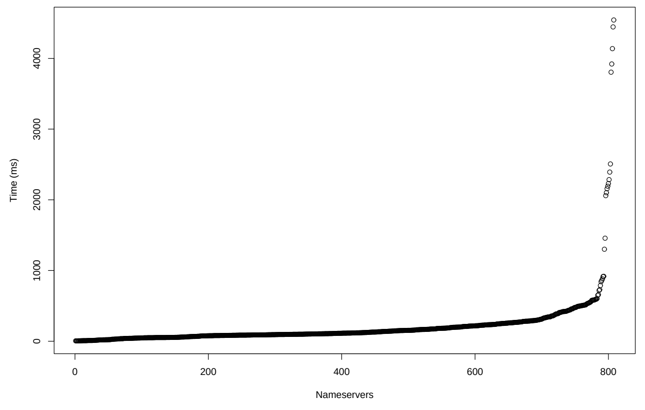

Fig. 7. Time for second query to NS1, asking NS2.

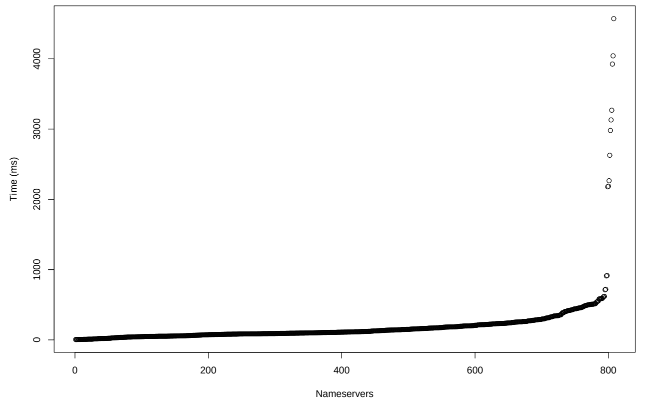

Fig. 8. Time for third query to NS1, asking NS2.

the lower end, only about 4% of the times are above 1s.

Figure 9 shows the actual round-trip time between the two nameservers. The data is taken from the times for the second query to NS2 subtracted by the times for the second query to NS1. This graph is interesting because it shows the same trend as the other graphs, meaning that there was a consistency in the times that lent to the validity of the data. Also, as can be seen from the few negative values, there is a small amount of fluctuation in the times, as should be expected. Because the negative values make up less than 6% of the values and their distance from zero is very small, it is clear that they are not significant problems with the data. We looked at these negative values for any times of a second or more to check for problems, but the lack of this meant that these negative numbers were simply the

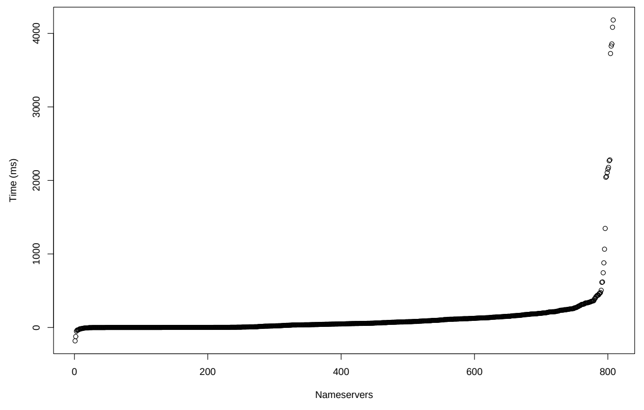

Fig. 9. Time between NS1 and NS2, based on the second queries to each server.

result of slight variances in the round-trip times and they could be counted as zeros.

Another aspect of Figure 9 that we needed to analyze was the values that are above the majority of the data. There are very few of them, helping to show that there are not any major problems with the data collected. Interesting though are the clusters of times around 2 and 4 seconds. We are currently still researching the cause of this clustering, but the current theory as of this writing is that they are related to timeouts that occured while we collected our data.

Having now established that our data appears valid, the most important feature of Figure 9 is strong grouping of times along a line from 0 to about half a second. This is the result that we were looking for, showing that the vast majority of round-trip times are well below the most common average time used for query timeouts, 5 seconds. Therefore, it is quite plausible that the default timeout value can and should be lowered in order to account for the current state of technology.

V. Conclusions

The data very clearly shows a concentration of times under one second. Current implementations of nameservers appear to have a timeout close to one or two seconds in between server queries, but this time varies. In addition, this is not a set default and the timeout grows exponentially with successive failed queries, making it farther and farther away from the bulk of the round-trip times that we found.

Although we are still researching it, the clustering around two and four seconds appears to show the effect of timeouts by having a similar trend as the bulk of the data, but a few seconds above. We believe that our data has shown that current default timeouts could be changed to be closer to the optimal timeout. The effects of the current disparity from the optimal time are seen when there is packet loss, in which case it is likely that the user may see delay of several seconds while trying to perform a lookup. Therefore, a shorter timeout of one second would limit this delay while keeping the number of redundant queries to a minimum.

REFERENCES

[1] E. Cohen and H. Kaplan, “Proactive caching of DNS records: Addressing a performance bottleneck,” in *Proceedings of the Symposium on Applications and the Internet (SAINT)*, January 2001.

[2] P. B. Danzig, K. Obraczka, and A. Kumar, “An analysis of wide-area name server traffi c: A study of the domain name system,” in *Proceedings of ACM SIGCOMM*, January 1992.

[3] T. Lee, B. Huffaker, M. Fomenkov, and k claffy, “The problem of optimization of dns root servers’ placement,” In Passive Measurement and Analysis Workshop’03, 2003., 2003. [Online]. Available: citeseer.ist.psu.edu/lee03problem.html

[4] R. Liston, S. Srinivasan, and E. Zegura, “Diversity in dns performance measures,” 2002.

[5] Z. M. Mao, C. D. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, and J. Wang, “A precise and effi cient evaluation of the proximity between web clients and their local DNS servers,” in *Proceedings of USENIX 2002*, June 2002.

[6] C. Wills and H. Shang, “The contribution of DNS lookup costs to web object retrieval,” Worcester Polytechnic Institute, Tech. Rep. TR-00-12, July 2000.

[7] P. V. Mockapetris, “RFC 1035: Domain names — implementation and specifi cation,” Nov. 1987. [Online]. Available: http://www.faqs.org/rfcs/rfc1035.html

[8] A. Kumar, J. Postel, C. Neuman, P. Danzig, and S. Miller, “Common DNS Implementation Errors and Suggested Fixes,” Oct. 1993. [Online]. Available: http://www.faqs.org/rfcs/rfc1536.html

[9] L. Bent, G. rey, and M. Voelker, “Whole page performance,” 2002. [Online]. Available: citeseer.ist.psu.edu/bent02whole.html

[10] D. Wessels, M. Fomenkov, and N. Brownlee, “Measurements and laboratory simulations of the upper dns hierarchy,” 2004. [Online]. Available: citeseer.ist.psu.edu/article/wessels04measurements.html

[11] Y. Sekiya, K. Cho, A. Kato, R. Somegawa, T. Jinmei, and J. Murai, “Root and cctld dns server observation from worldwide locations.” [Online]. Available: citeseer.ist.psu.edu/567083.html

[12] C. J. Brandhorst and A. Pras, “Dns: a statistical analysis of name server traffi c at local network-to-internet connections,” 2005. [Online]. Available: www.simpleweb.org/nm/research/results/publications/pras/2005-Eunice-Brandhorst.pdf

[13] N. Brownlee, K. Claffy, and E. Nemeth, “DNS measurements at a root server,” in *Global Internet 2001*, November 2001, presentation slides are available at http://www.caida.org/outreach/presentations/ietf0112/dns.damage.html.

[14] P. V. Mockapetris, “The internet and identifi ers,” in *Proceedings of ACM SIGCOMM*, August 2005. [Online]. Available: http://www.sigcomm.org/sigcomm2005/slides-Moc.pdf

[15] Z. Bar-Yossef and M. Gurevich, “Random sampling from a search engine’s index,” 15th International World Wide Web Conference. [Online]. Available: http://www2006.org/programme/fi les/xhtml/3047/3047-baryossef-xhtml.html