Hrithik Bandaru
17XJ1A0508

DNS Performance and the Effectiveness of Caching

Jaeyeon Jung, Emil Sit, Hari Balakrishnan, *Member*, *IEEE*, and Robert Morris

Abstract

A trace conducted at MIT found that 23% of the lookups received no answer and that these lookups account for more than half of all the DNS packets.13% of all lookups result in answers which indicates an error condition and the major reason for this is missing inverse mappings or Name server (NS) records that point to an in appropriate hosts. While most successful answers are received in at most two to three retransmissions, failures today cause a much larger number of retransmissions and, thus, packets that traverse the wide area. This paper discusses the detailed analysis of traces of the domain name system(DNS) which is responsible for the inverse mapping and the associated TCP traffic collected on the internet links of the MIT laboratory for Computer Science and Korea Advanced Institute of Science and Technology(KAIST), This paper is focused on how clients at these institutions interact with the wide area domain name system by focusing on client perceived performance and on effectiveness of DNS caching. They discuss the two factors widely believed to contribute to the scalability of DNS which are hierarchical design around administratively delegated name spaces and the aggressive use of caching by analysing the traces at both MIT and KAIST.

Introduction

The Domain name system (DNS) is like a phonebook of the internet. Where we humans use it to access information by mentioning their domain name and the browser interacts through the internet protocol address (ip address).DNS translates the domain name of website we need to access to it's ip address in a similar way we use a phonebook to acquire someone's phone number through their name. The DNS converts the domain names to network locations so that the web browser can load internet resources. The DNS is very critical to the internet as it maps the names to network locations. It has to be distributed globally and must be highly scalable and offer good performance under high load. The DNS must perform efficiently since it has to provide low latency to the client while minimizing the amount of wide area network resources it consumes.

Terminology :-

- A lookup mechanism for translating a domain name for a client.
- A query refers to a DNS packet sent to a DNS server.
- A response refers to a packet sent by the DNS server in reply to the query packet.
- An answer is a response from a DNS server that terminates the lookup, by returning either the requested name to record mapping or and error indication

Caching is the process of storing copies of files in a cache so that they can be accessed more readily during the time of need. DNS servers cache DNS records for faster lookups, CDN servers cache in a similar way to reduce latency. Caching in DNS is normally not size constrained since the items being stored are small, comprising generally of close to 100 bytes for each entry. Every asset record is terminated by the time set by the originator of the name. These termination times are called TTL values.

DNS maps a human-readable host names to a computer friendly IP addresses. It also provides other valuable information about the domain or host, including reverse maps from IP addresses to host names and mail-routing information. There are two types of resource records there is address records (A records) and name server records (NS records). An address specifies the name's IP address and an NS record specifies the name of a DNS server that is authoritative for a name. The namespace needs to be made hierarchical to be able to scale. Clients that make recursive queries are known as stub resolvers. DNS lookups can be performed by any device. Remote DNS data is locally cacheable to improve performance. The actual lookups ask questions to the DNS on behalf of the clients and the answers are obtained from authoritative servers. Answers are stored for future reference in the cache.

There are four types of DNS servers these four kinds of servers work in harmony together to return the final answer that is the IP address for a specified domain by the client. The four types include: Recursive servers, Root nameservers, Top level domain (TLD) nameservers and Authoritative nameservers. The client is usually a stub resolver which is simple resolver which is built into the operating system on the platform used by the client.

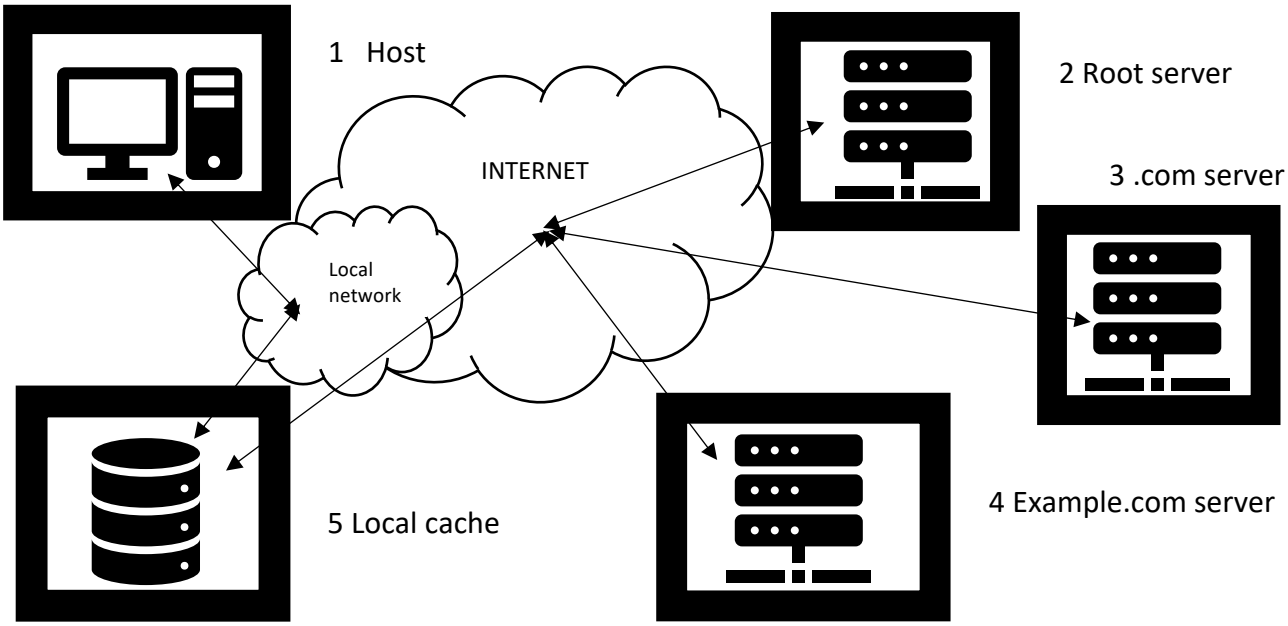

Fig. 1.

Steps involved in a DNS lookup sequence refer to Fig. 1 :-

1. The host asks the for address of a website www.example.com.
2. Local DNS server doesn't know it and, asks root, Root server refers to a .com server.
3. The .com server refers to a .example.com server.
4. The example server responds with an address.
5. The local server caches response and responds to host.

Here, We can see that a lookup may involve many queries and responses. The queries of a lookup ask data but from different DNS servers and all responses apart from the last response are referrals. As we can see this happen in Fig. 1 the packets in all the steps from 1-4 are just part of one lookup and multiple packets received as answers are just referrals which lead us to the final answer.

Description

It is generally accepted that two components add to the adaptability of DNS: progressive structure around authoritatively designated namespaces, and the aggressive use of caching. The two variables look to decrease the load on the root servers at the top of the namespace hierarchy , while fruitful caching tries to reduce client perceived delays and WAN bandwidth usage. How viable are these variables? In this paper, they cautiously investigate three network traces to consider this inquiry.

The only large scale study on DNS performance prior to the year 2000 was by Danzig and Danzig's study found that a large number of implementation errors caused the DNS to consume 20 times more WAN bandwidth than necessary. DNS implementations and usage pattern have changed since then .Content distribution networks (CDNs) and popular Web sites now use DNS as a level of indirection to balance load across servers, provide fault tolerance, or route client requests to servers topologically close to the clients. Because cached DNS records limit the efficacy of such techniques, many of these multiple-server systems use time-to-live (TTL) values as small as a few seconds or minutes. A TTL is a setting that tells the DNS resolver how long to cache a query before requesting a new one.

The percentage of DNS packets in the wide-area packets is downward trend as it reduced from 14% in 1990 in a study conducted by Danzig to 3% in 1997 in a study of the MCI backbone by Thompson suggests that DNS caching is working well. However, these results should be put in perspective by considering them relative to network traffic as a whole. Thompson's study also showed that DNS accounts for 18% of all unidirectional traffic stream with unique source and destination IP addresses, port numbers, and IP protocol fields. These contemplations make an intensive investigation of the viability of DNS caching is particularly significant. Thus,

this paper helps us in exploring DNS performance and scalability, it focuses on the performance DNS clients perceive in terms of latency and failures and how varying the TTL values and the degree of cache sharing impact caching effectiveness. It also focuses on evaluating the effectiveness of DNS caching on how important it is for it's scalability and it's unanticipated uses of DNS.

One of the major motivations behind this research was to calculate the ratio of the DNS lookups to TCP connections in the wide area network. This ration will essentially give us the DNS cache hit ratio. They needed to calculate the number of DNS queries per lookup as this quick calculation will give us an idea about the effectiveness of the caching at suppressing wide area traffic and also the percentage of DNS lookups which do not get an answer. The paper also focuses on the effect of varying TTLs and degrees of cache sharing on DNS cache hit rates by conducting trace driven simulations.

Solution Description

The study was based on three separate network traces. The first two traces were conducted at the link that connects the MIT LCS and AI labs to the rest of the internet in January and December 2000. At that time there were 24 internal subnetworks sharing the router, used by over 500 users and 1200 hosts. The final trace was conducted at one of the two links connecting KAIST to the rest of the internet which was in MAY 2001. At that time there were 1000 users and more than 5000 hosts connected to the KAIST campus network.

Collection Methodology

After careful filtration of the traffic collected at the points, the data ser was produced which was useful for the purpose. From previous studies we have learnt that TCP traffic comprises the bulk of wide area traffic. TCP flows can be hence viewed as a major driving workload for DNS lookups. For this reason they specifically collected outgoing DNS queries and incoming responses and outgoing TCP connection start, end packets for connections starting inside the networks. The trace connection points cannot collect all DNS activity as queries answered by caches inside the traced networks do not appear in the traces. Since we corelate the lookups that require wide area queries to be sent as the driving work load of TCP connections we can still use this data to conclude about the performance and effectiveness of caching. All packets were rewritten to anonymize the source IP addresses of hosts inside the traced network by mapping each IP address using a keyed MD5 cryptographic hash function to an essentially unique anonymized one.

The software used for collection was derived from Minshall's tcpdriv utility. It can collect traces directly or postprocess them after collection using a tool such as tcpdump. We extended the tcpdriv to support the anonymization scheme described above for DNS traffic.

Analysis Methodology

The analysis of the data obtained was mainly to calculate the number of referrals involved in a typical lookup and the distribution of the lookup latency.

The latency is calculated by resolving a lookup. by maintaining a sliding window of lookups seen in the last sixty seconds. An entry is added for each query packet from an internal host with a DNS query ID different from any lookup in the time window. We can find a corresponding lookup in the window if there is an incoming response packet. If the response packet is the answer, the time difference between the original packet and the answer packet is the lookup latency. If the incoming response is not a response we increase the number of referrals by one and wait for the final answer to arrive. This method estimates the list of servers contacted for iterative lookups, but not recursive lookups.

Data Summary

Roughly 50% of the DNS lookups at MIT are not associated with any TCP connection. Approximately, 15% of these A lookups are for name servers. Also, roughly 10% of all lookups are the result of an incoming TCP connection. We suspect that the remaining 20% of these correspond to UDP flows but, unfortunately, our data does not include any record of UDP flows. Approximately 20% of TCP connections fall into the second class. Here, the dominant cause is the establishment of FTP-data connections. Other causes include hard-coded and hand-entered addresses.

The major motivation behind the collection of this data was to find the ratio of DNS lookups to TCP connections in the wide area network. From the data collected we can find the ratio of the total TCP connections to the valid A answers which is about 7.28 for MIT in January 2000, 4.91 for MIT in December 2000 and 7.75 for KAIST in May 2001.These numbers show that there is DNS cache hit ratio of 80-87% for all the three traces. This hit ratio is not particularly high because this involves caching done by web browsers when they open multiple connections to the same server.

All the patterns and values discussed below were drawn from plotting the cumulative distribution functions of different attributes against different factors that were believed to have an impact.

Client Perceived DNS Performance

From the data obtained by using the above methods they have calculated the median latency of each trace and they are 85ms for MIT January 2000, 97ms for MIT December and 42ms for KAIST. Referrals adversely impact the latency. Cached NS records benefits the latency because it reduces the load on the root servers severly.

Lookups that result in no answer, and lookups that require retransmissions in order to elicit an answer. This is interesting because the total number of queries is a lot larger than the number of lookups, from the obtained data we see that the average number of query packets in an answered query is 1.27 for MIT January 2000, 1.2 for MIT December and 1.2 for KAIST May 2001. We can calculate the ratio r from the formula $r = 1+ (Q – X – L)/I$. Where Q stands for the total number of queries , L is the total number of lookups in the trace and I is the number of iterative lookups in the trace and X is the number query packets corresponding to retransmission of recursive lookups. By calculating the value of r from the data we get the values as 2.40 for MIT January 2000, 2.57 for MIT December 2000, and 2.36 for KAIST in May 2001. r is substantially larger than the average number of query packets for an answered lookup because retransmissions account for a significant fraction of all DNS packets seen in the wide-area Internet.

By the cumulative distribution of the number of retransmissions for answered and unanswered lookups we see that each lookup that elicited zero referrals generated about five times as many wide-area query packets before the querying name server gave up. We can also see that over 99.99% of all the answered lookups needed one or two retransmissions and over 90% of the lookups needed no retransmissions. The percentage of zero referral lookups increased from 5.5% to 9.3% and the percentage of these causing more than five retransmissions increased from 10% to 13% from January 2000 to December 2000 at MIT.

By looking at the cumulative distribution functions (CDFs) of the number of query packets generated for the nonzero referrals and loops categories of unanswered lookups we see that a large fraction of the traced DNS packets are caused by lookups that end up receiving no answer. Over 63% of the traced DNS query packets were generated by lookups that obtained no answer and the corresponding number for MIT January 2000 is 59%. Some of these were required to overcome packet losses on Internet paths. The typical loss rates are between 5% and 10 and the number of redundant DNS query packets observed in our traces is substantially higher than this.

Between 10% and 42% of lookups result in a negative answer and most of these errors are either NXDOMAIN or SERVFAIL. NXDOMAIN error is usually caused when the requested name does not exist and SERVFAIL usually means that the server is supposed to be authoritative and does not have a valid copy of the database, it may also suggest that the server has run out of memory. Most of the errors are caused by inverse lookups for IP addresses with no inverse mappings. Most lookups were accounted for by a relatively small number of names and each looked up a large number of times. Presumably the NS records for these names were misconfigured.

Large number of `NXDOMAIN` duplicate responses suggests that negative caching may not be working as well as it could be. The actual cause of the most of the negative responses is unknown but it may be because of the typos of the real names, the reverse lookup for an address that does not have a reverse mapping, reverse blacklist lookups, number of misconfigured servers forward queries for the name loop-back, instead of handling it locally and there can be several such reasons. The distribution of names causing a negative response follows a heavy-tailed distribution as well and that could be the reason hit rate of negative caching is also limited.

It is observed that 15% to 18% of lookups contacted root or gTLD servers and the percentage slightly decreased between January and December 2000 and this was probably caused by an increase in the popularity of popular names, which decreased DNS cache miss rates. It is also observed that the load on root servers has been shifted to gTLD. The gTLD servers were serving more than half of all top-level domain queries by the end of 2000. Between 15% and 27% of the lookups sent to root name servers resulted in negative responses. Name servers could reduce root load by refusing to forward queries for unqualified host names. The number of these lookups resulting in negative answers remains roughly the same during 2000, but because of the shift to gTLDs, the relative percentage of these lookups has increased.

Effectiveness of Caching

To understand how popularity of names varies in the traces a graph is plot where the access counts as function of the popularity rank of a name. previous studies that showed that Web object popularity follows a Zipf-like distribution, we represent the access count as a function a / x^{α}, where α is termed the popularity index. A line fit would closely fit the tail and would give us the value of α by calculating the negative of the slope of the line which is equal to 0.91. The tail behaviour changes when names are aggregated according to their domains. This will give us α values greater than one indicating that the tail falls of a little faster than in a fully qualified one.

By plotting the fraction of the most popular distinct names and the cumulative fraction of answered lookups accounted for by the corresponding most popular names we see that about 10% of the most popular names account for 68% of total answers and a large number of names are accessed only once, which suggests that a significant number of top level domain queries will occur regardless of the caching scheme.

The cumulative distribution of TTL values for A and NS records show that NS records tend to have much longer TTL values than A records. This helps explain why only about 20% of DNS come from a root or gTLD server. The load on the root or gTLD server would be five times the current load if NS records had a lower TTL values as all the DNS look up traffic would have gone towards them.

The TTL distribution of names, weighted by the fraction of TCP connections made to each name. From this we can see that it is indeed the case that shorter-TTL names are more frequently accessed this is because we already know that DNS-based load-balancing makes sense only for popular sites and the fraction of access to relatively short TTL values has doubled since their observation from January to December 2000 at the MIT trace which suggests that there is an increase in deployment of DNS-based servers.

The benefits of having a shared or a per client DNS caching of A records trace driven simulations of cache behaviour under different conditions were conducted. Two databases were formed from the DNS answers obtained at the trace. The name database maps every IP address appearing in an A answer to the domain name. The TTL database maps each domain name to the highest TTL appearing in an A record for that name.

After the collection of data the following steps were taken to conduct the simulation :-

1) Randomly divide the TCP clients appearing in the trace into groups of size s. Give each group its own simulated shared DNS cache, as if the group shared a single forwarding DNS server. The simulated cache is indexed by domain name, and contains the TTL for that cached name.
2) For each new TCP connection in the trace, determine which client is involved by looking at the inside IP address and let that client's group be . Use the outside IP address to index into the name database to find the domain name n that the client would have looked up before making the TCP connection.
3) If n exists in g's cache, and the cached TTL has not expired, record a hit. Otherwise, record a miss.
4) On a miss, make an entry in g's cache for n, and copy the TTL from the TTL database into the n's cache entry.

At the end of each simulation run, the hit rate is the number of hits divided by the total number of queries.

This simulation has a bunch of weakness and they are; that multiple domain names may map to the same IP address, as sometimes occurs at Web hosting sites and this may cause the simulations to overestimate the DNS hit rate. The simulation also assumes that each client belongs to a single caching group, which may not

be true if a client uses multiple local forwarding DNS servers and since DNS clients typically query servers in a strictly sequential order this wouldn't impact the simulation significantly.

From the simulation it is established that the average cache hit for a per connection cache for December 2000 is 71% and 89% if all the 1216 traced clients at the MIT in January 2000. The fact that domain name popularity has a Zipf-like distribution explains this. But just by 10 or 20 clients sharing a cache obtained most of the benefits. A small number of names are very popular and hence a small degree of cache can take advantage of that and large number of names are accessed by a very small number of clients. References to these large number of less popular names are likely to be sequential references and per client or per application caches built into the web browser takes advantage of this.

The TTL values in DNS records affect cache rates by limiting the opportunities for reusing cache entries because of this the algorithm is slightly modified. All the TTL values are set to specific values instead of the ones taken from the actual DNS responses. There are 2 distributions observed where the cache is shared among all the clients and the other where the cache is shared among groups of 25 clients. 25 because we previously found that such a number would get most of the benefits. The benefit of caching is achieved when the TTL values a small number of minutes. Single clients looking up the same server multiple times in quick succession, a pattern probably produced by web browsers loading multiple objects from the same page or users viewing multiple pages from the same website cause most cache hits.

From these results we can see that the practise of using low TTL values in DNS based server does not greatly impact the hit rates. There would be twice the DNS traffic if we eliminated all but per client and there would be four times the DNS traffic if we eliminated all A record caching. This evidence favours recent shifts toward more dynamic uses of DNS, such as mobile host location tracking and sophisticated DNS-based server selection. The function of caching to reduce client latency can almost be handled entirely by per-client or per-application caching.

Decreasing the TTL values on A records or NS records is not a good idea as it would increase the load on the servers by a factor of five and it would also severely harm the scalability of DNS.

Conclusion

23% of lookups receive no answer and these lookups account for more than half of all traced DNS packets in the wide area. Answers for lookups are received in at most two to three retransmissions and failures cause a much larger number of retransmissions and that is the reason why more than half of all DNS packets account for the lookups which receive no answer. About 13% of all lookups result in a negative response as it appears to be caused by missing inverse mappings or NS records that point to inappropriate hosts. Over 25% of the queries sent to the root name servers result in such failures.

The trace driven simulations conducted yielded two significant results. Reducing the TTLs of A records to as low as a few hundred seconds has little adverse effect on hit rates and the cacheability of NS records efficiently partition the name space and avoiding the overloading of any single name server is the factor behind the scalability of the DNS.

This paper analysed the client perceived performance of DNS including the latency to receive answers, the performance of DNS, including the latency to receive answer, the performance of the DNS protocol , the prevalence of failures and errors and the interaction with root/gTLD servers. Trace driven simulations were conducted to analyse the effectiveness of DNS caching as a function of TTL and degree of cache sharing.