IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 2, January 2023
Impact Factor: 7.301

Domain Name System

Mr. Pradeep Nayak[1], Nandan M R[2], Sushma K N[3], Srusti K[4], Diya H B[5]
Assistant Professor, Department of Information Science and Engineering[1]
Students, Department of Information Science and Engineering[2,3,4,5]
Alva's Institute of Engineering and Technology, Mijar, Mangalore, Karnataka, India

Abstract: *The DARPA Internet's name service is provided via the Domain Name System (DNS). One of the biggest name services now in use, it provides a uses a special mix of hierarchies, caching, and datagram access and is composed of a highly diversified community of servers, users, and networks. This article explores the concepts that underlie the initial design of the DNS in 1983, explains how these concepts have evolved into the present implementations and usages, highlights notable surprises, successes, and failings, and makes predictions about how the DNS may develop in the future. Nameserver delegations, the foundation of the Domain Name System (DNS), generate intricate and delicate dependencies between names and nameservers. In this work, we report the findings of a comprehensive DNS survey and demonstrate how these dependencies result in a very vulnerable naming system. It examines the efficiency of name caching and the calculation of retransmission timeouts, demonstrates how algorithms to boost DNS's resilience result in disastrous behavior when servers fail or when specific implementation faults are triggered, explains the paradoxically high proportion of wide-area DNS packets, and assesses the effects of flaws in various DNS implementations. It demonstrates how DNS performance would only be slightly enhanced by negative caching in a network with suitably configured name servers. It ends by urging a fundamental shift in how we design and implement name servers and distributed applications in the future.*

Keywords: DNS

I. INTRODUCTION

Around 1982, it was noticed that the HOSTS.TXT method for publishing the mapping between host names and addresses was having or was about to have issues. This led to the creation of the DNS. A basic text file with the name HOSTS.TXT is sent to all hosts on the Internet via direct and indirect file transfers from a host at the SRI Network Information Center (SRI-NIC), where it is centrally maintained. The issues were that the update process was centralized, which did not fit the trend toward increasingly decentralized management of the Internet, and that the file was growing too huge, increasing the expense of its dissemination. The DNS, which converts host names to IP addresses, is essential for maintaining the reliability of Internet services and applications. However, security threats posed by DNS's design are challenging to foresee and manage. In order to resolve a name to its IP address using the delegation-based architecture used by DNS, the names of the servers in charge of that name must first be resolved. There are intricate relationships among DNS servers as a result of the resolution of these server names, which in turn depends on other name resolutions. Overall, multiple servers are involved in the resolution of a single name, and any of them being compromised can have a significant impact on the security of DNS and the applications that rely on it. The main functions of DNS, a distributed, replicated name service, are to locate daemons for electronic mail transfer, map host names into corresponding Internet addresses, and translate Internet addresses into hostnames [12, 11]. Its name space is structured as an unbalanced tree, with 1,000,000 leaf nodes denoting individual computers and 16,000 unique inside nodes known as domains as of late 1991 [8]. In essence, each of these devices is a DNS client. A resolver is a piece of DNS client software, and there is at least a dozen of them available [7]. To fix different problems, some of these implementations have been re-released.

II. DNS DESIGN

The DNS's fundamental design tenets required that it: Offer at the very least the same data as HOSTS.TXT. Give permission for distributed database maintenance. Do not impose clear size restrictions on names, name components, name-related data, etc. Interact with as many different contexts as you can, including the DARPA Internet. Give

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 2, January 2023
Impact Factor: 7.301

respectable performance. The following derivative restrictions were included: Only if the system offered extensible services could the cost of implementation be justified. The system should be capable of encapsulating various name spaces and independent of network topology. The system should not try to impose a certain OS, architecture, or organizational style on its users in order to be widely accepted. This concept was relevant from issues with case sensitivity to the notion that the system should work for both big timeshared hosts and solitary PCs. In principle, we aimed to allow as many diverse implementation structures as possible while avoiding any externally imposed limits on the system.

2.1 The Architecture

Name servers and resolvers are the two main categories of DNS active components. Name servers are information repositories that respond to inquiries with the data they have available. Resolvers connect to client programmers and contain the algorithms required to locate a name server that holds the data the client is looking for. Depending on the requirements of the environment, these functions may be merged or separated. Centralizing the resolver function in one or more unique name servers for an organization is frequently beneficial. This structure allows for the sharing of cached data and enables less powerful hosts, such as PCs, to rely on specialized servers' resolving services without the need for a resolver on the PC.

2.2 The Name Space

Each node in the variable-depth tree that makes up the DNS internal name space has a corresponding label. The label combinations on the path from a node to the tree's root make up a node's domain name. Each octet in a label can be any 8-bit value. Labels are strings of octets that can be any length. The root is assigned the zero-length label. Case is not taken into consideration during name space searching operations (for operations described at this time) (assuming ASCII). As a result, the names "Paul," "Paul," and "PAUL" would all be synonymous. By effectively forbidding the construction of brother nodes with labels that differ only in case but have the same spelling, this matching rule. For the purpose of promoting the use of DNS to encrypt already-existing structured names, the DNS specification avoids providing an uniform printing rule for the internal name format. Applications are free to behave differently from how configuration files in the domain system display names, which are character strings separated by dots. Host names, for instance, follow internal DNS regulations, so VENERA.ISI.EDU has four labels (the null name of the root is typically left out). Mailbox names with the format USER@DOMAIN (or more generally local-part@? organization) encode the text to the left of the "a" in a single label, possibly including "." and apply the dot-delimiting DNS configuration file rule for the portion that comes after the @. For file names, etc., similar encodings may be devised.

III. DNS OVERVIEW AND THREATS

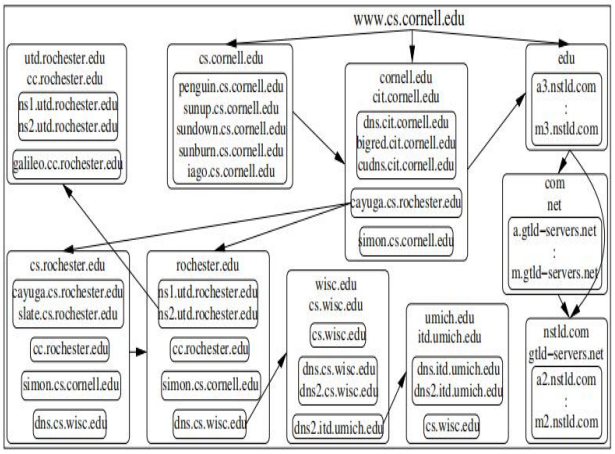

Figure 1: Due to its delegation-based architecture, DNS shows intricate dependencies between nameservers. For instance, a nameserver at umich.edu is indirectly used by the domain name www.cs.cornell.edu. Dependencies are shown in the diagram with arrows. For the sake of clarity, self-loops and unnecessary dependencies have been removed.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 2, January 2023
Impact Factor: 7.301

Domains are the hierarchically divided, non-overlapping sections of the DNS namespace. For example, cs.cornell.edu is a sub-domain of cornell.edu, which in turn is a sub-domain of the top-level domain Edu, which is under the global root domain.

A group of nodes referred to as the domain's authoritative nameservers service names inside that domain. The root nameservers and top-level domain authoritative nameservers are at the top of the DNS hierarchy (TLDs). Generic TLDs (gTLDs) like.com,.Edu, and.net and country-code TLDs (ccTLDs) like. Uk, tr, and.in make up the top-level domain namespace.

For name resolution, DNS employs a delegation-based architecture [6, 7]. Clients resolve names by following an authoritative chain of nameservers that descends from the root, then the nameservers for the TLD, and finally the nameservers for the query name. For instance, the parent domains Edu, cornell.edu, and cs. Cornell. Edu's authoritative nameservers are used to resolve the name www.cs.cornell.edu. The delegation interdependencies for www.cs.cornell.edu are shown in Figure 1. The resolution of this name is dependent on twenty nameservers in addition to the top-level domain nameservers, only nine of which are in the cornell.edu domain. The namespace of Cornell is indirectly governed by several nameservers that are not within Cornell's administrative control. In this instance, umich.edu is dependent upon umich.edu, which is dependent upon wisc.edu, which is dependent upon rochester.edu. While cayuga.cs.rochester.edu is the nameserver that Cornell directly trusts to serve its namespace, it has no control over the nameservers that Rochester.edu trusts.

IV. PERFORMANCE ISSUES

The caching method, the RPC retransmission timeout algorithm, and the algorithm for choosing alternate name servers are the obvious problems when evaluating the performance of any replicated, distributed name service. These mechanisms were purposefully left undefined when DNS was first introduced since effective solutions were not yet understood. Maybe this was done to keep the description of mechanism and policy separate. Nine years after DNS's initial deployment, practical fixes for each of these problems are now available; this section outlines one of them.

4.1 Caching

The results to their successful queries are cached by DNS name servers. Responses stating that a name is invalid or stating that the requested resource record is not connected to the query name are often not cached. Negate caching is the practise of saving poor words or the absence of a resource record in order to efficiently respond to future queries. Each resource in a DNS domain is given a time-to-live by the person in charge of the domain so that other name servers can cache it for that amount of time. Every time a cached value is sent to another server, its time-to-live is shortened by the amount of time it spent there. One to six days is the typical time-to-live number, while zero turns off caching. Keep in mind that caching is handled by the name server and not by the client resolver code that users include in their programmes. For a location with hundreds or thousands of computers, like a college campus, this has significant ramifications.

A program's resolver sends a query to a name server each time it does so. All the requests on well-run websites pass through a small number of regional name servers. One of ten nearby name servers, for instance, receives queries from each computer at our university. The more central servers there are, the more likely it is to find a working name server. Ten is perhaps too many, as it makes it more likely that our site will make additional requests on the wide-area network.

4.2 Retransmission Algorithm

As previously indicated, Berkeley UNIX includes the most popular resolver and name server, which are currently in their seventh public release (BIND 4.8.3). This software, which is a good example of an implementation with few bugs, is accessible via public ftp from ftp. UU. net. In order to avoid the additional messages and delay required to establish and tear down a TCP connection, almost all DNS transactions use the UDP unreliable datagram protocol. Retransmissions are now the user program's problem. The Berkeley resolver and name server's retransmission techniques are outlined in this section.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 2, January 2023
Impact Factor: 7.301

4.2.1 Resolver Algorithm

For each request, the Berkeley resolver checks a maximum of three name servers. This can be increased and is parameterized. The query is sent to the first of the three servers by the resolver, who then waits for a timeout. If neither server responds, it sends the message to the second server and waits. If neither server responds by the time the timer expires, it sends the message to the third server and waits. The timeout interval is doubled and the three servers are tried sequentially again if it times out once more without obtaining a response from any of the three servers.

4.2.2 Name Server Algorithm

For the purpose of answering requests, the Berkeley name server keeps track of up to 16 different server addresses. It queries the addresses in the order that the potential servers are arranged by predicted response time to previous inquiries. The servers can be cycled through up to three times, barring infinite looping. The default retransmission timeout is set to the greater of 4 seconds or twice a server's anticipated service time. It increases the retransmission timeout once each cycle is finished, although it is always restricted to 45 seconds. Looping might happen if the answer to a query contains a reference to another name server or a suggestion to try a different canonical name. An alias for the query name, like castor, is a canonical name. USC.edu is another name for girted. The server is only allowed to accept up to 20 referrals or 8 canonical name changes. After executing the cycle of 16.3 = 48 queries, the server responds with the status severalise if there are no referrals or canonical name substitutions. Only inquiries for the set of root name servers are an exception to this rule. These are referred to as system inquiries, and they are run hourly until they are successful. If the server is still attempting to resolve the initial question, it discards retransmitted inquiries.

4.3 The Net Effect

To answer a query it receives, a Berkeley name server will make up to three requests to each address of another name server. A single query to an unreplicated, unreachable server would result in around 9 query packets because a resolver could query up to 3 name servers.

System queries are given special consideration and are retried once per hour until they succeed since a name server must be able to connect to at least one root server. This suggests that, for instance, if only one of the two addresses in the priming file is working and this server is unavailable, a total of 3x24 = 72 requests would be made to it over the course of a day.

V. CURRENT IMPLEMENTATION STATUS

On the DARPA Internet, the DNS is in use. The implementations or ports listed in [RFC 10311 range from the widespread support included with Berkeley UNIX to those for IBM-PCs, Macintoshes, LISP machines, and fuzzballs [Mills 8 81]. Older hosts continue to use the HOSTS.TXT protocol, but the DNS is the suggested mechanism. A recent measurement [Stahl 871] indicated approximately 5,500 host names in the current HOSTS.TXT, whereas over 20,000 host names were available via the DNS. Hosts available through HOSTS.TXT represent an ever-dwindling subset of all hosts.

5.1 Root Servers

A resolver can search "downward" from domains it can already access thanks to the DNS's basic search algorithm. The root node and the top of the local domain are often pointed to servers by "hints" that are configured into resolvers. As a result, if a resolver can connect to any root server, it can access the entire domain space. If the resolver is in a network that is isolated from the rest of the Internet, it can at least access local names. Although root server availability is a critical resilience problem and root server activity monitoring offers insights into DNS usage, resolvers query root servers less as they accumulate cached knowledge about servers for lower domains. A query is typically sent to each root server once every second, with higher rates experienced when other root servers are offline or experiencing issues. Although there has been a general increased trend in query rate, day-to-day and month-to-month comparisons of load are more influenced by modifications to implementation techniques and timeout adjustment than by an increase in the clientele. One problematic release of well-known domain software, for instance, caused average loads to exceed five

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 2, January 2023
Impact Factor: 7.301

times the norm for extended periods of time. Currently, we predict that using less aggressive retransmission and greater caching in various resolver implementations would eliminate 50% of all root server traffic.

5.2 Berkeley

The University of California, Berkeley supplied UNIX support for the DNS, in part as distributed systems research and in part out of necessity owing to the expansion of the campus network [Dunlap 86a, Dunlap 8 6b]. The Berkeley Internet Name Domain (BIND) server is the outcome. Berkeley is used as an example of a sizable delegated domain, even though it is undoubtedly more advanced and experienced than most. Berkeley set up machines with all their network applications fully dependent on DNS for performing network host and address resolution as the first organisation on the DARPA Internet with BIND. In the spring of 1985, Berkeley began to set up workstations on its campus that were reliant on the name server.

VI. SUCCESSES

6.1 Datagram Access

Given the unexpectedly poor performance of the DARPA Internet, it was successful and likely necessary to employ datagrams as the preferred means of reaching name servers. It turns out that the limitation to approximately 512 bytes of data is not a problem. Compared to TCP circuits, performance is significantly superior, and OS resources are not constrained. The need to create and improve retransmission techniques, which are already pretty well defined for TCP, is the single clear disadvantage to datagram access. Resolvers that were constructed to the point of functioning but whose authors lost interest before tuning or systems that imported well-known versions of code but do not track tuning updates generate a lot of unneeded traffic.

6.2 Additional Section Processing

A name server is free to include any additional information in the response it chooses, as long as it fits in a single datagram, when responding to a query, in addition to the information it used to do so. The goal was to avoid incurring a large additional communication cost by enabling the responding server to foresee the following logical request and answer to it beforehand. For instance, the root servers always provide the host's address when returning the name of the host, if the other information will require the host address. Research demonstrates that this feature reduces query traffic in half.

6.3 Mail Address Cooperation

An agreement to use organizationally structured domain names for mail addressing and routing was reached by representatives of the CSNET, BITNET, UUCP, and DARPA Internet groups. 4.3

VII. DNS REPLICATION

DNS's performance is impacted by the frequency of domain replication on different servers in two ways: With more domain replication, servers are more resilient to failures on their own. Unfortunately, the level of replication increases the severity of DNS issues related to zero answer counts and recursion from forwarding lists. Our traces revealed that thousands of problematic name servers sent packets to the 1S1 root name server is two network addresses. This leads us to believe that some flawed implementations probably flood all domain replicas with the same problematic queries. In the May and September traces, it responded to 75% and 25% of these queries for approximately 1,149 and 665 thousand wide-area packets, respectively. Keep in mind that the server was only operational for nine hours in the latter trace. We believe that many of the same queries were being handled by loops and zero count issues on the other six root servers. We don't have any data to back this up, though. NSFnet reported 11 million DNS packets daily in May 1991. On the day we conducted our trace, the 1S1 root name server sent and received roughly 1 million wide-area packets. The root servers sent or got 7 million DNS packets if the other six root servers sent and received roughly the same number of packets.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 2, January 2023
Impact Factor: 7.301

VIII. ERROR DETECTING SERVERS

Systems must be built to keep track of their own behaviour and look for malfunctioning parts. The reason why most implementations ignore this component is because it reduces performance in most cases. In actuality, DNS uses around 20 times more bandwidth than it needs to. Servers should be able to discover common implementation faults in other name servers and resolvers in order to boost Internet reliability. The expense of techniques to identify misbehaving implementations would be quickly recovered. Not that we advocate packet tracing for each one, mind you. Of course, doing so reveals amazing things, such as the fact that every ten minutes, an IBM PC/RT at Stoneybrook sends a bogus empty DNS response packet to the 1S1 root name server.

8.1 Feigning Death

If a name server occasionally pretends to be dead, it can see problematic behaviour related to server failure, bad root server priming caches, and inappropriate retransmission timeout algorithms. It does not have to act completely rigour mortis. Rather, it chooses a name from a list of names after 200 or 300 inquiries. It decides not to respond to inquiries for this name for the following five minutes. It stores all the tries for this name in a data structure during this time. It is likely that this domain does not handle server failure adequately if a server tries to query the name more than two or three times, or if multiple servers from the same domain request it. The server is equipped to log network addresses, domain names, and other information.

8.2 Bean Counting

Typical name servers take too long to respond. One dumb question can be asked a thousand times, and it will always spit forth the same response. Not everyone is this patient. Name servers ought to keep track of the network addresses, request and response packet counts, and timestamp of the most recent query in a data structure that is indexed by query name. With ageing timestamps, this data structure can be garbage collected. A trash collected entry should be written to the log file if the packet count is excessive. The remote resolver or name server may be experiencing issues if a live entry's packet count increases too much and the response code of this query shows the inquiry has zero replies.

8.3 Policing

Our name server should launch a process once per day that reads the log and sends emails to the managers of computers or domains that implement names incorrectly. Wide-area network switches could implement a similar functionality. These switches had the ability to monitor name traffic and look for oddities. Policing will rapidly solve names, something that patience and hope have failed to do.

IX. CONCLUSION

It was possible to delay the need for a new system and weaken the quantitative justifications for the DNS by making changes to the HOSTS.TXT scheme. The administrative, communication, and support workload for the entire community have probably not yet been lessened by the DNS. We do believe that the need to disperse functions was inevitable. The main evaluation factors must be this requirement, as well as the additional functionality and potential for future services. The authors defend the DNS from their point of view. While there are many decisions we might make differently if we had a second chance, the following principal pieces of advice would have been helpful when we were just getting started: Although caching can function in a diverse environment, it should also include mechanisms for saving unfavourable outcomes. It is frequently more challenging to get a new function added to a system than it is to get a function removed. Not every member of a community would switch to a new service; rather, some would continue using the old one, some would switch to the new, and others would support both. The bad result of this is that as additional features are introduced, all functions get more complex. Due to the complexity of DNS, it is difficult to define and bind trust connections, and a flaw in a minor nameserver may have far-reaching effects. Trust ties can alter without being noticed, even if the name owners are careful and assess the level of dependencies at the time of name creation. The overuse of transitive trust is to blame here [13]. Delegating nameservers creates a dependency tree, and issues like failure resilience and independent administration allow the resulting dependency graphs to expand significantly and alter rapidly. A small trusted computing base (TCB) is generally desirable since smaller TCBs are

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 3, Issue 2, January 2023
Impact Factor: 7.301

simpler to secure, audit, and administer. This is a widely acknowledged principle in computer security. According to our assessment, the TCB in DNS is substantial and can comprise. As faulty name servers and resolvers are replaced, the proportion of wide-area network traffic caused by DNS will decrease, supposing no vendor publishes another catastrophic update bug. Wide-area DNS traffic would be reduced by a factor of twenty and possibly more if, for instance, all implementations adopted the Berkeley name server and resolution techniques (see Section 2) in their entirety. Analysis of our traces shows that negative caching is not required in a name server-run Internet. However, negative caching can act as a barrier between a wide-area network and dangerous or seriously flawed programmes. However, we don't think the entire Internet will be made up of well-functioning resolvers and name servers. As outdated implementations are updated, vendors, operating system updates, and bridges between implementations will release flawed new versions.

REFERENCES

[1] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the Domain Name System Security Extensions. Request for Comments 4035, Mar. 2005.
[2] C. Huitema and S. Weerahandi. Internet Measurements: The Rising Tide and the DNS Snag. In Proc. of ITC Specialist Seminar on Internet Traffific Measurement and Modeling, Monterey, CA, 2000.
[3] Internet Systems Consortium. BIND Vulnerabilities. http://www.isc.org/sw/bind/bind-security.php, Feb. 2004.
[4] J. Jung, A. Berger, and H. Balakrishnan. Modeling TTL-based Inter net Caches. In Proc. of IEEE International Conference on Computer Communications, San Francisco, CA, Mar. 2003.
[5] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS Performance and Effectiveness of Caching. In Proc. of SIGCOMM Internet Measurement Workshop, San Francisco, CA, Nov. 2001.
[6] P. Mockapetris. Domain Names: Concepts and Facilities. Request for Comments 1034, Nov. 1987.
[7] P. Mockapetris. Domain Names: Implementation and Specification. Request for Comments 1035, Nov. 1987.
[8] P. Mockapetris and K. Dunlop. Development of the Domain Name System. In Proc. of ACM SIGCOMM, Stanford, CA, 1988.
[9] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, and L. Zhang. Impact of Confifiguration Errors on DNS Robustness. In Proc. of ACM SIGCOMM, Portland, OR, Aug. 2004.
[10] K. Park, V. Pai, and L. Peterson. CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups. In Proc. of Symposium on Operating Systems Design and Implementation, 2004.
[11] V. Ramasubramanian and E. G. Sirer. The Design and Implementation of a Next Generation Name Service for the Internet. In Proc. of ACM SIGCOMM, Portland, OR, Aug. 2004.
[12] A. Shaikh, R. Tewari, and M. Agarwal. On the Effectiveness of DNS-based Server Selection. In Proc. of IEEE International Conference on Computer Communications, Anchorage, AK, Apr. 2001.
[13] K. Thompson. Reflflections on Trusting Trust. Comm. of the ACM, 27(8), Aug. 1984.
[14] C. E. Wills and H. Shang. The Contribution of DNS Lookup Costs to Web Object Retrieval. Technical Report TR-00-12, Worcester Polytechnic Institute, July 2000