DNS Project Rapport

Handed in 25. April, 2014, by team no. 9

Bardur Simonsen	20091058@post.au.dk
Thor Ottesen	10358@post.au.dk

Electrical and Computer Engineering
Aarhus University
Finlandsgade 22, 8200 Aarhus N, Denmark

Abstract

This rapport will describe the history, fundamentals and use of Domain Name Service, How it works and the termonology used, furthermore it will also describe the configuration and setup of our BIND installation also refered to as our Prototype. The rapport will briefly cover alternatives to DNS.

Contents

Chapter 1

Introduction

This rapport is primarily about the Domain Name System (DNS), it will document the principles of DNS, what it is, what DNS is used for and how it works. Furthermore this rapport will also contain information about how the world of computer networks worked before DNS was created. The predecessor to DNS and the shortcomings and developments that created a need to handle naming and identification of computers in a more sophisticated way and a more scalable manner. Lastly this rapport will document our "prototype".

The prototype we have choosen to built is a name server that is installed and configured for a hypothetical scenario where a company needs to host their own name server in order to better serve the company and provide a better and more stable DNS service where certain names are blacklisted in order to better secure the network and work efficiency.

This rapport is relevant because it's about one of the fundamental technologies of the internet and it describes some fundamental design principles of scalable and highly available computing. Not knowing how to properly scale your software can be a costly and damaging experience and should be avoided at all cost.

This rapport divided into three parts, it will begin by telling the reader about the age before DNS and what incited the need for DNS, it will give an introduction to DNS and explain the various concepts of DNS like name resolution, secure DNS, etc. The second part will demonstrate our prototype, the user scenario, configuration and performance. The third part will provide an conclusion and perspective of out prototype and about DNS in general.

Chapter 2

Domain Name System

A time long gone, were the concept of a unified global communications network, that could make the worlds combined knowledge accessible at the fingerprints of every person, was still the essence of science fiction. Something magical was brewing.

In the basements of DARPA's office building a man named Robert Tylor, was intrigued by the concept of an interconnected packet network system. Set out on a quest to realize this very concept, with the help of Lawrence Roberts from MIT, he build ARPANET.

ARPANET started as a link between the University of California, Los Angeles (UCLA) and the Stanford Research Institute, the link was established 22:30 on October 29, 1969. Even though that ARPANET's beginning was humble it herald a new age and ARPANET became the technical core of what would become the internet. ARPANET was initially based on the Network Control Protocol (NCP) but on January 1st. 1983 also known as flag day. The NCP protocol was replaced with the more powerful and flexible family of TCP/IP protocols and marking the era of the modern internet.

TCP/IP completely changed the way ARPANET worked. It introduced the IP stack and attributes like IP Address, IP Prefix and gateway. The IP address is a 32bit integer that identifies a given node on the network, the IP Prefix signifies the number of most significant bits of the IP address that is used to identify the network that the respective node is a part of. The gateway is the router that serves as an access point to another network.

Soon after ARPANET was first launched, it started to grow, and it started to grow quickly. At this time the nodes where still identified and accessed by using their numerical addresses and with the ever increasing number of nodes connected to the network a need for a simpler and memorable way of identifying the nodes arose.

The first method of alleviating the problem of numerical addresses was the HOSTS.TXT file. The staff at the Stanford Research center compiled a text file that mapped intelligible names to numerical addresses of computers on ARPANET. The host file was then routinely updated and circulated, but the internet's rapid growth made the centrally maintained host file a bottleneck and a new system that could scale with the growth of the internet was needed.

2.1 Intruduction to DNS

In 1983, Paul Mockapetis designed the Domain Name System (DNS). DNS is meant to replace the HOST.TXT file, so you can access websites without remembering the IP address. Figur X shows a screenshot of a browser that has accessed Google's search services by using Google's IP address 74.125.136.104.

All servers can be reached by using their IP or domain (altrough some software or configurations don't support accessing it's services by IP) but it is much easier to remember how to spell Google then it is to remember four numbers between 0 and 256 and thats the reason why DNS is an fundamental and essential service in the modern internet.

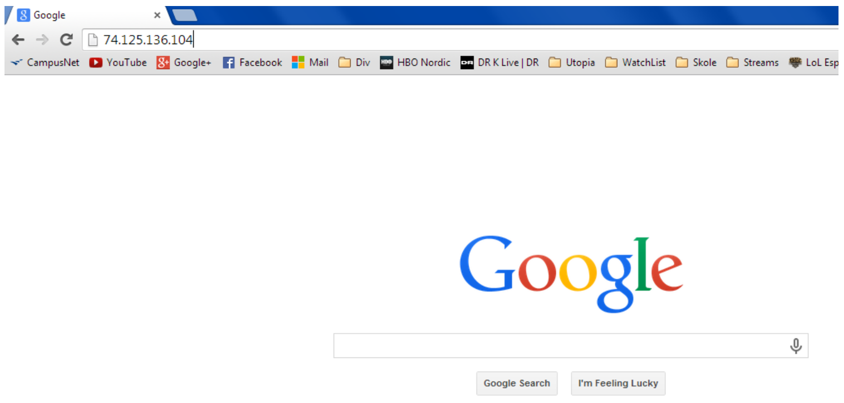


Figure 2.1: Screenshot of accessing Google by IP

DNS is essentially a system to distribute the HOST.TXT file on a massive scale. It works by breaking up the Fully Qualified Domain Name (FQDN.) into a hieratical structure delimited by the top level domain, (TLD), domain and subdomains. This structure is illustrated in the figure below.

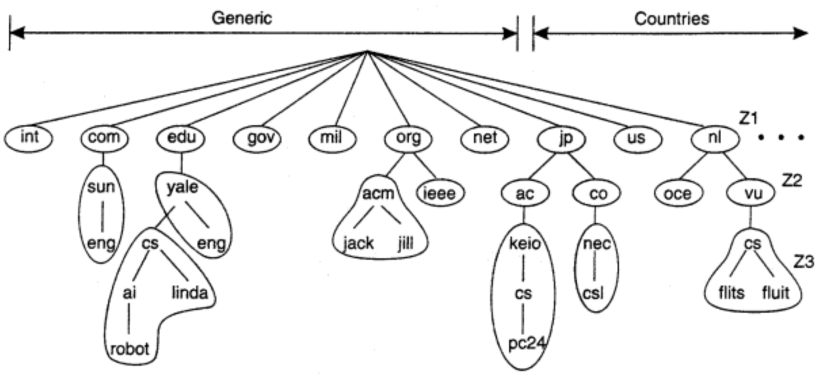


Figure 2.2: A illustration of the DNS topology!

The first level in DNS is the root servers, represented by the dot suffix in the FQDN, the next level is the TLD's like .net, .com, .dk, etc. then comes the domain and subdomains. There are 13 root domain servers which is known by all the domain servers. Each domain server also knows of the domain servers on the level below it which it is connected to. From fig 2.2 you can see that the domain server .org knows of .acm and .jeee. In this way all domains can be found from the root by a chain of domain servers knowing of the levels below them.

2.2 Name Resolution

When we have a name that is needed to be translated to an address then it can be done iterative or recursive. When it is done iterative then the client calls the root witch only know the address of the server on the layer below. It then returns that address to the client. Then the client calls that server which gives the address of the server below. This process is repeated until a server returns the address of the server with the full name.

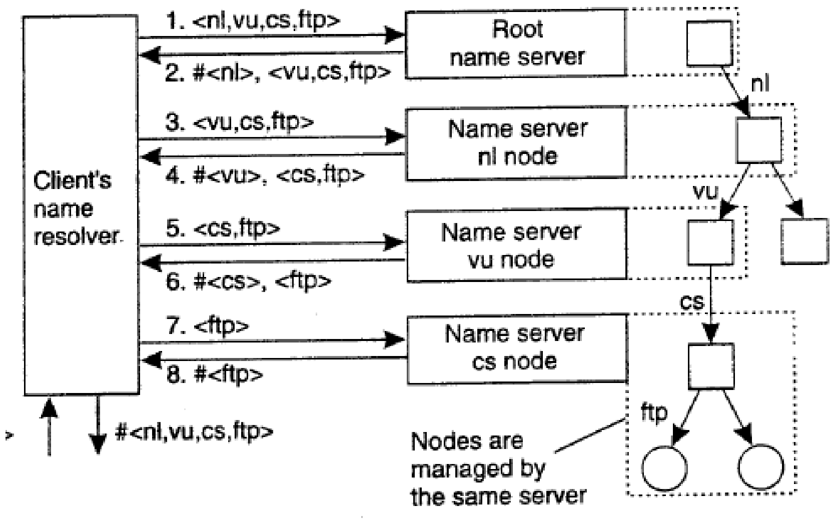


Figure 2.3: Iterative Domain Name Resolution

When the process is done recursively then the client calls the root. The root does not call the client back but calls the server on the layer below. When the bottom layer server is found it calls up the chain to the root with the answer. The root then returns the answer to the client.

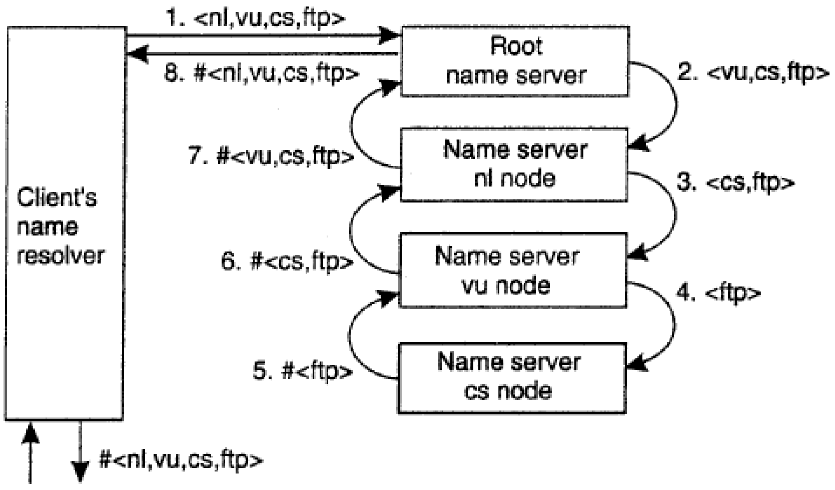


Figure 2.4: Recursive Domain Name Resolution

The downside of recursive approach is that it is more performance heavy on the servers because it needs to handle the whole request. So not all servers, support recursive name resolution. One of the benefits with recursive name resolution is that it is better for caching the resolutions. When the result is passed up the chain then the servers up the chain can easily caches the resolution. Then next time the server gets a request for the name, it doesn't need to call the server below it to get the resolution. With recursive name resolution the client only need to send one request for the resolution compared to one for every subdomain.

2.3 DNS Security

DNSSEC is an encryption system for DNS that is used to validate the data that is received from a DNS server. The reason for this is that a spoof DNS server with intercepts the communication could send corrupted data to the client. DNSSEC uses a key pair, which can encrypt/decrypt each other's messages. Data is send with a digital signature that is encrypted by the sender. Then the receiver decrypts the signature and compares it with the massage. The only drawback is that if the is given a fake public key then this is not secure. To ensure that the key is valid there is DNSSEC trust relationship. This works by other DNS servers having a trust anchor for a remote DNS serve. This anchor is the public key of the remote server and now the server with the anchor can validate massages for all the zones the remote server is authoritative for.

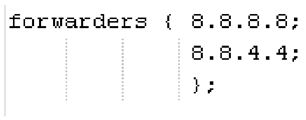
Chapter 3

About your prototype

For this project we have chosen the hypothetical scenario of a medium to large company that wants to run their own DNS. The company choose to run their own DNS primarily their ISP's DNS isn't performing well enough and because they wanted to restrict access to certain domains that contains information adverse to the companies IT security, efficiency and in general domains that are adverse to company morale. We have chosen to use BIND is our name server and Ubuntu 14.04 LTS as our operating system. We have chosen this combination because it is a proven and tested solution that has stood the test of time. BIND is the defacto standard name server and linux provides excellent performance and stability.

3.1 Installation and configuration

To installed BIND we have used Ubuntu's package manager and installed the bind package. We used the following Linux commands to install BIND Sudo apt-get install bind After we installed BIND, we configured it to forward and cache DNS queries to another DNS server. We carefully choose the DNS server to forward to, after a series of comprehensive and rigorously benchmarking tests. We benchmarked all public DNS servers with the DNS benchmarking tool NameBench from Google. The results of the tests showed that Googles own public DNS was the best performing DNS server. To configure bind to forward it's queries to Googles public DNS server and added the following configuration to binds configuration file, filename.



```
forwarders { 8.8.8.8;
             8.8.4.4;
             };
```

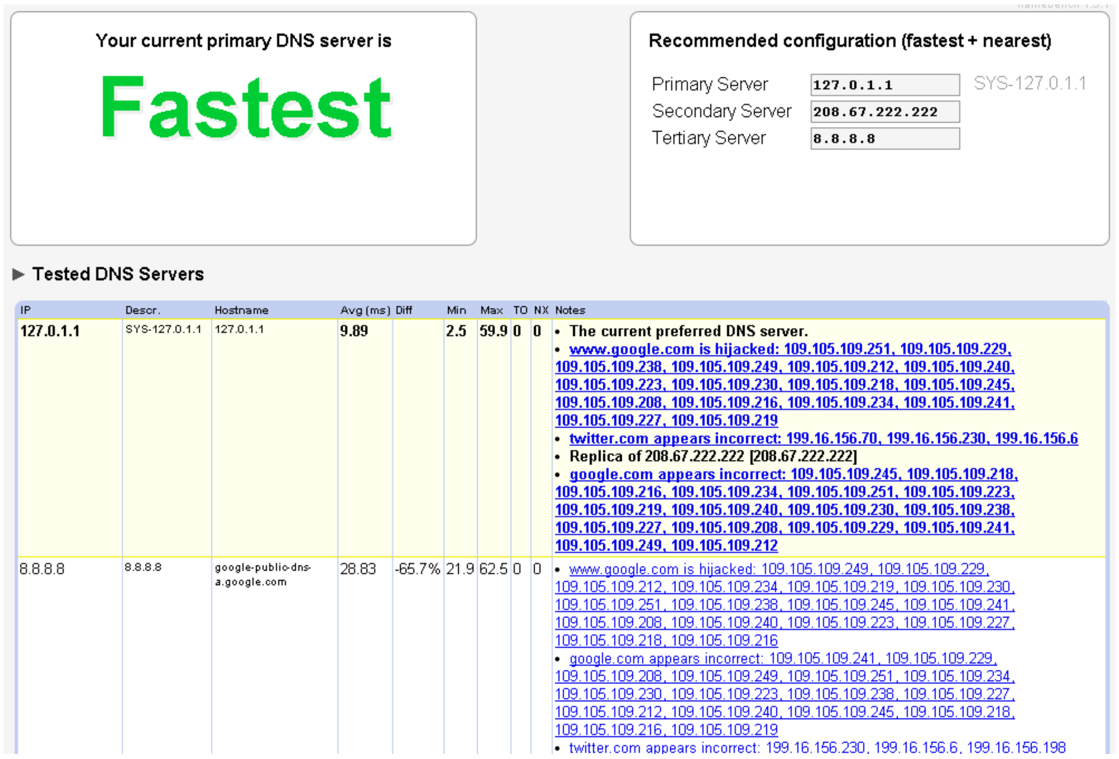
Another goal of running our own DNS server was to have the ability to restrict access to certain domains, to accomplish this we created a blackhole file. A blackhole is a feature in BIND that restricts access to certain domains. To configure a blackhole we created a fake zone file and added make certain domains point to that fake zone file.

```
zone "google.com"  type master; file "dummy-block"; ;
zone "donkeyporn.com"  type master; file "dummy-block"; ;
```

the above configuration adds a fake zone to the domains google.com and donkey-porn.com. this ensures that company employees don't waste company time googling or watching donkeyporn.

3.2 Benchmarking

To verify if the performance goal of hosting our own DNS server has been accomplished we have run a series of benchmarking tests to find the fastest DNS server. We have used the tool called NameBench created by Google configured to test our server and compare it with OpenDNS and Googles Public DNS



Your current primary DNS server is

Fastest

Recommended configuration (fastest + nearest)

Primary Server	127.0.1.1	SYS-127.0.1.1
Secondary Server	208.67.222.222	
Tertiary Server	8.8.8.8	

▶ Tested DNS Servers

IP	Descr.	Hostname	Avg (ms)	Diff	Min	Max	TO	NX	Notes
127.0.1.1	SYS-127.0.1.1	127.0.1.1	**9.89**		**2.5**	**59.9**	**0**	**0**	• **The current preferred DNS server.** • **www.google.com is hijacked: 109.105.109.251, 109.105.109.229, 109.105.109.238, 109.105.109.249, 109.105.109.212, 109.105.109.240, 109.105.109.223, 109.105.109.230, 109.105.109.218, 109.105.109.245, 109.105.109.208, 109.105.109.216, 109.105.109.234, 109.105.109.241, 109.105.109.227, 109.105.109.219** • **twitter.com appears incorrect: 199.16.156.70, 199.16.156.230, 199.16.156.6** • **Replica of 208.67.222.222 [208.67.222.222]** • **google.com appears incorrect: 109.105.109.245, 109.105.109.218, 109.105.109.216, 109.105.109.234, 109.105.109.251, 109.105.109.223, 109.105.109.219, 109.105.109.240, 109.105.109.230, 109.105.109.238, 109.105.109.227, 109.105.109.208, 109.105.109.229, 109.105.109.241, 109.105.109.249, 109.105.109.212**
8.8.8.8	8.8.8.8	google-public-dns-a.google.com	28.83	-65.7%	21.9	62.5	0	0	• www.google.com is hijacked: 109.105.109.249, 109.105.109.229, 109.105.109.212, 109.105.109.234, 109.105.109.219, 109.105.109.230, 109.105.109.251, 109.105.109.238, 109.105.109.245, 109.105.109.241, 109.105.109.208, 109.105.109.240, 109.105.109.223, 109.105.109.227, 109.105.109.218, 109.105.109.216 • google.com appears incorrect: 109.105.109.241, 109.105.109.229, 109.105.109.208, 109.105.109.249, 109.105.109.251, 109.105.109.234, 109.105.109.230, 109.105.109.223, 109.105.109.238, 109.105.109.227, 109.105.109.212, 109.105.109.240, 109.105.109.245, 109.105.109.218, 109.105.109.216, 109.105.109.219 • twitter.com appears incorrect: 199.16.156.230, 199.16.156.6, 199.16.156.198

Figure 3.1: Screenshot of benchmark test rapport

The above figure show that our DNS server is the fastest server by almost 60 percent. This clearly shows that running your own DNS server is beneficial to the performance.

Chapter 4

Conclusion

DNS is a great naming system that is decentralized, massively scalable and flexible. Its predecessor the “Host file” would not be able to support a network as large and dynamic as the internet. Bind was easy to setup and use and gives a lot of options to configure the network access. It proved to be a flexible and powerful platform at served our needs excellent. Our prototyped worked great, it gave us a much faster and responsive DNS and the ability to restrict access to certain domains. We can conclude that the prototype was a great success.

Bibliography

[1] Distributed systems: Principles and Paradigms by Andrew S. Tanenbaum

[2] http://answers.oreilly.com/topic/2892-how-does-public-key-cryptography-work-in-dnssec/