ITONK Project
DNS

Handed in 25th of April, 2014, by team no. 8

Stephan Thordal Larsen	11072@iha.dk
Cong Thanh Dao	20073860@iha.dk
Kasper Duong	08754@iha.dk
Jakob Toftegaard Holst	10494@iha.dk
Sam Luu Tong	10898@iha.dk

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

Abstract

The purpose of this document is to describe the technical aspects of a Domain Name System, abbreviated as DNS, and the protocol they implement. This includes the security considerations and a brief description of the DNS Security Extension.

A case study is included to uncover benefits of using a DNS server in a school. The case study also sets ground for a description of the steps needed to get a DNS server running.

Contents

Chapter 1

Introduction

The report is about the Domain Name System (DNS). In the report the following subjects will be touched:

- DNS fundamentals

- DNS name resolution

- DNS security extensions

- BIND DNS server

The relevance of this report is justified by showing some of the fundamentals regarding the DNS. The DNS plays its role by supporting the internet infrastructure and by providing a distributed and fairly robust mechanism that resolves host names into IP addresses and IP addresses back into host names. It will also provide decent demonstrations of touching DNS elements throughout the report.

The structure of the report follows the above list of subjects. In the end of the report, a prototype will be presented followed by a discussion and conclusion of the DNS project.

Chapter 2

Technology
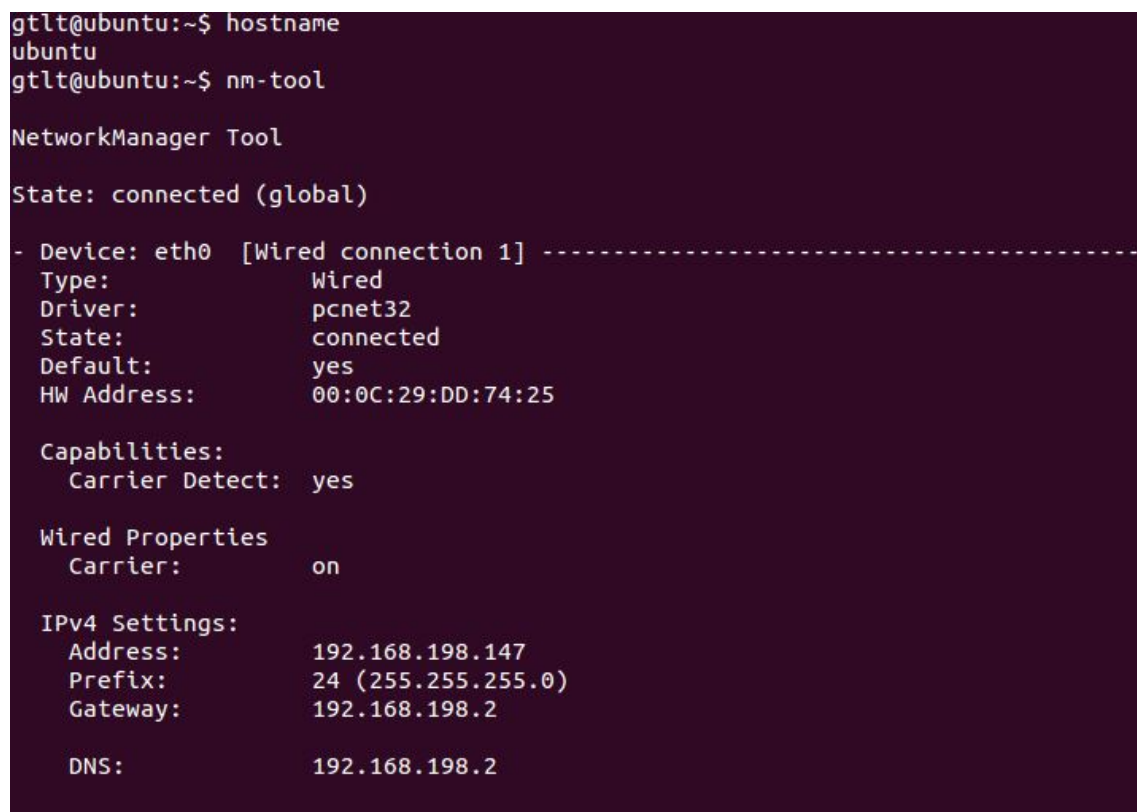
2.1 DNS Fundamentals

The DNS is used to resolve host names, making it possible to search for a name instead of an IP-address. This ensures that you can make a search for e.g. `www.google.com` and not get an error, even if they have changed IP-address at some point.

2.1.1 Demonstration of DNS fundamentals - Linux

The *hostname* command shows the hostname of the system - in this case, it would just be **ubuntu**.

The *nm-tool*, which is demonstrated below, is an utility that provides information about NetworkManager, devices, and wireless networks.

By running the command in a Linux terminal, we get the following output:

```
gtlt@ubuntu:~$ hostname
ubuntu
gtlt@ubuntu:~$ nm-tool

NetworkManager Tool

State: connected (global)

- Device: eth0  [Wired connection 1] -----------------------------------
  Type:              Wired
  Driver:            pcnet32
  State:             connected
  Default:           yes
  HW Address:        00:0C:29:DD:74:25

  Capabilities:
    Carrier Detect:  yes

  Wired Properties
    Carrier:         on

  IPv4 Settings:
    Address:         192.168.198.147
    Prefix:          24 (255.255.255.0)
    Gateway:         192.168.198.2

    DNS:             192.168.198.2
```

Figure 2.1: Screenshot of running commands *hostname* and *nm-tool*

4

The *Address* field tells us about the current internal IP-address for the device. The internal IP is an IP assigned for the device that is connected to a router.

The *Prefix* tells us the number of significant bits used to identify a network. Subnet mask 255.255.255.0 has a prefix of 24 bits, which tells us that the first 24 bits identifies the network, and the last 8 bits identifies the specific machine.

The *Gateway* field tells us about the IP-address for the current host router. In general, a gateway is a network point that acts as an entrance to another network.

The *DNS* fields tells us which IP-address they connect to and make a lookup when accessing the internet.

2.1.2 Demonstration of DNS fundamentals - in Windows

It is also possible to view those fields in Windows, which can be done by opening a command window in Windows and then run the command *ipconfig /all*.

This will demonstrate about pinging a webserver. Ping is a computer network administration utility used to test the reachability of a host on an Internet Protocol (IP) network and to measure the round-trip time for messages sent from the originating host to a destination computer. We will experiment with `www.dr.dk` and then type in the IP-address in a webbrowser - it <u>will</u> show the same result:
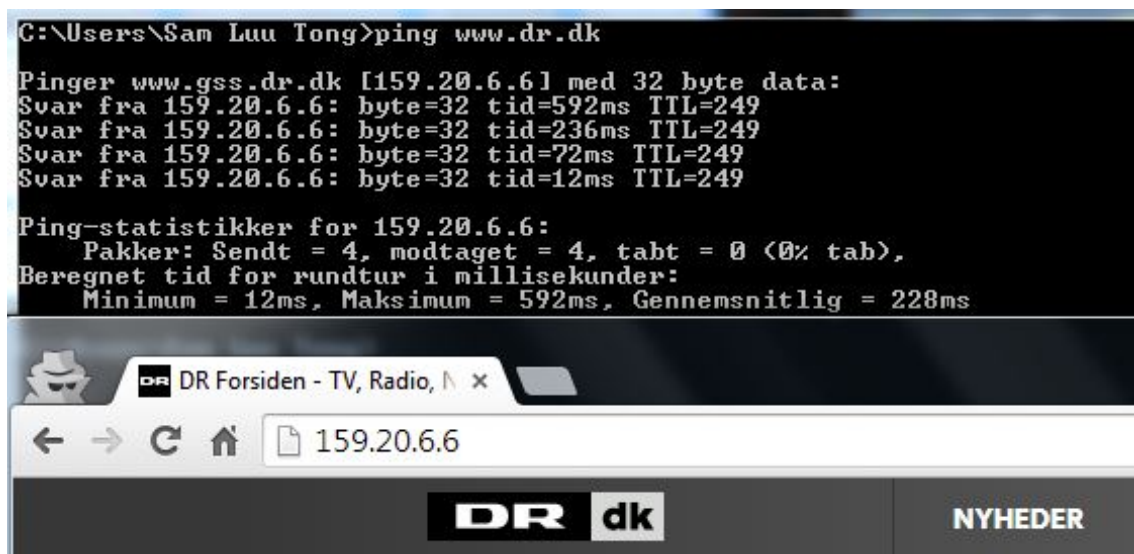

Figure 2.2: Screenshot of pinging www.dr.dk

So whether we type `www.dr.dk` or we type 159.20.6.6 doesn't make a difference - only that `www.dr.dk` probably is easier to remember!

2.1.3 Host Lookup Table - Background and demonstration
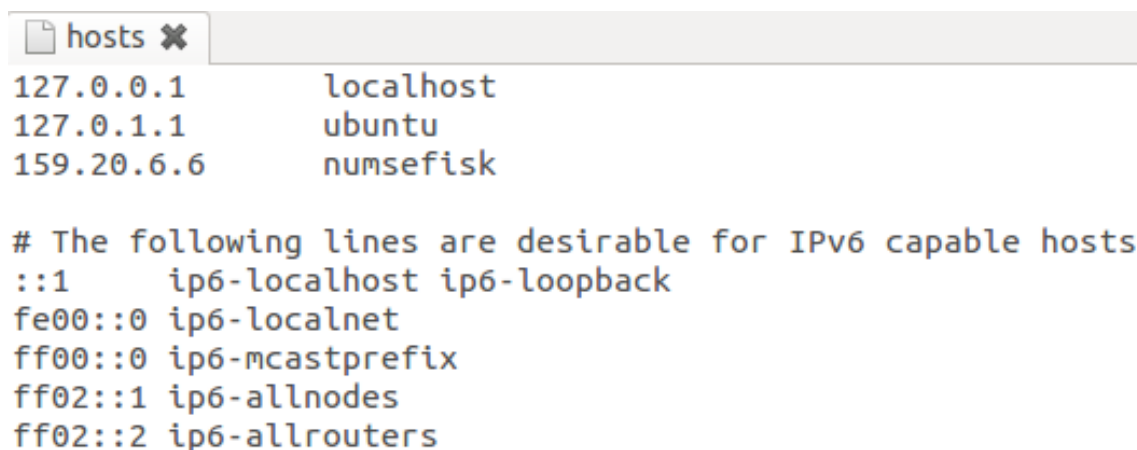
2.1.3.1 The Background

Before DNS was fully deployed, a woman, **Peggy Karp**, created a lookup table that mapped all of the network resources in one text formatted file. This formatted file was a simple ASCII text file called "HOSTS.TXT", and the table contained all of the hostnames and their related IP addresses.

Operators would install this file on their local server, which would then gain the capability to perform the requisite lookups locally and enable the computer to find resources out on the larger network without a lot of overhead. Whenever an operator added a new machine to the network, they would complete an email template with the appropriate information, send it off to the appropriate people at Stanford Research Institute (SRI) who would compile all of the changes and include them in the next release of HOSTS.TXT and store the new file on a globally available FTP server.

Operators would retrieve the updated versions on a regular basis and install them on their local servers. The first version of this table was distributed in *1971-1972*. While this arrangement worked well for a number of years, but it suffered from one systemic problem – it wasn't scalable as the network grew in popularity and new hosts were added. Therefore, the size of HOSTS.TXT grew in direct relationship. For each host added, HOSTS.TXT added a new record, and if the operators did not update their records on a regular basis, HOSTS.TXT would grow out of date which led all sorts of confusion. On the brighter side, it led the engineers of the time to come to the conclusion that a new structure would have to be put into place to replace HOSTS.TXT.

2.1.3.2 Demonstration

The Hosts.txt file still exists (in Linux under the folder /etc) and can be used to locally map hostnames to IP-addresses. To demonstrate this, the hostname "numsefisk" has been mapped to the IP-address of `www.dr.dk`, which is 159.20.6.6:


```
127.0.0.1        localhost
127.0.1.1        ubuntu
159.20.6.6       numsefisk

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Figure 2.3: The change "numsefisk" made in Hosts.txt

By running a browser and type "numsefisk", you will be redirected to www.dr.dk, or as an alternative, you could just ping "numsefisk" to test the changes.

It's worth knowing what happens first - is the HLT looked through before your primary DNS server is queried? The answer can be found in host.conf, because this central file controls the resolver setup, which services to use and in what order. By opening the host.conf, the order is "host,bind", and therefore the HLT is looked through before the BIND DNS.

2.1.4 Top Level Domain

To understand how DNS work, you have to consider some elements.[1] If you look at an internet address, it is made of the www.second-level-domain.com, where the .com is the top level domain. The TLD defines in which TLD the router shall look for in the second-level-domain name. The .com TLD is a generic TLD (gTLD), which states for which purpose they are to be used. It's worth to mention that Top Level Domains for different countries exist [2] like .dk, .jp or .de.

---

[1] https://archive.icann.org/en/tlds/
[2] These are called country codes Top Level Domain (ccTLD)

2.1.5 Fully Qualified Domain Name

A domain name can be either fully qualified(FQDN) or not. The definition of a FQDN is its unambiguity, that is, it is unique, and there is only one interpretation of the domain. Hence also the dot at the end of a FQDN to define it is so e.g.
`http://www.dns-sd.org./TrailingDotsInDomainNames.html`.

2.1.6 DNS zone and records

A DNS zone is a portion of a Domain Name System, just like the area code in a post district. The zone has been delegated administrative responsibility for the DNS.

When a domain name is looked up, a DNS zone returns a record, possibly a DNS "A" record. This depends on which record is needed to be returned considered the requested address. Records are data stored in the zone files of the DNS. An "A" record is a "Address record" that returns a 32-bit IPv4 address, most commonly used to map hostnames to an IP-address of the host.

2.2 Name Resolution

When we want to access a specific host via a webbrowser, we usually type in the host name to locate an identity. The host name uniquely identity an entity, which can be identified with an IP address. In short, a name resolution maps host names to its corresponding IP addresses.

So when a client wants to access any entries, it communicates with name servers which contains entries of host names and their corresponding IP addresses. This communication can vary depending on what type of DNS query is used; recursive and iterative queries.

2.2.1 Recursive Resolution

When a client wants to query for an entry, it requests to a DNS server, then do all the work of finding the entity and respond back to the client.

During the process, a DNS server might request other DNS servers to fetch the entity, i.e. when a client request for a hostname like example.com, it first looks up into its resolve.conf file to identity which DNS server it will make a query.

The DNS Server will then look through its own table (cache), and if not found it will ask the Root Server. The Root Server will return a list of servers, that is responsible for handling .COM domains (gTLD).

The DNS Server will choose one of the .COM gTLD Server and then query for the example.com, and if it exists, it will reply back and the DNS Server will respond back to the client with the IP-address that corresponds to the hostname.

2.2.2 Iterative Resolution

With an iterative query, the client will make a query to a DNS Server but this time, the server won't respond with the final answer unless the entity is found within its own table (cache). Instead it will respond back to the client with a referral to a Root Server. This process will continue on with communicating between the client and other servers till it's found, hence an iterative name resolution.

2.2.3 Iterative vs Recursive

When a browser makes a query, it will make a DNS query to a Resolver on the operating system (client). The Resolver has an algorithm that will identify if a certain IP-address is likely to be requested again. It will store it (caching). Likewise, with DNS caching, a caching name server will store DNS query results for a period of time depending on how each server is configured. This will improve the load and response time compared to Iterative resolution.

Caching on iterative resolution will increase the load and response time as the communication will be between client and server iteratively.

2.3 DNS Security Extension

The original design of DNS did not include security, which makes it vulnerable for hacking. There are several ways to integrate security directly in the DNS, One of them is DNSSEC that attempts to add security, while maintaining backwards compatibility.

2.3.1 Threads

There are several threads in DNS a query/response transaction. One of the simplest threads against DNS are various forms of packet interception: monkey-in-the-middle attacks, eavesdropping on requests combined with spoofed responses that beat the real response back to the resolver, and so forth.

Another interesting method of hacking is DNS spoofing or DNS cache poisoning attack. The DNS spoofing is a computer hacking attack into DNS server's cache database and change the name servers IP-addresses. By changing the IP-address, the user's traffic will be diverting to the hackers phishing site.

2.3.2 DNSSEC

Domain Name System Security Extensions (DNSSEC) is a digital signature, which is designed to prevent hackers from changing the DNS process and direct users onto their own website to commit phishing. A long-term solution to this vulnerability is an end-to-end deployment of a security code. It is not a technology that encrypts data, but it tells you about the validity of the address you are visiting. DNS entry is added a key and a signature that determines the validity of this key. Therefore, it does not alter the existing DNS protocol, but builds on it.

2.3.3 Signed zones

It is necessary to exploit it to the fullest by taking DNSSEC in use at each stage of the DNS process. Zones that implement DNSSEC are called signed zones, because they include digital signatures for resource records in their zone files, served by DNSSEC-aware authoritative name servers. DNSSEC incorporates a chain of digital signatures in the DNS hierarchy. On each level in the hierarchy, the owner has their own signature generating key.

For instance, the iha.dk has the following layers:

- Root-zone: ICANN who controls the Root server, they should have the DNSSEC in place.

- .dk-zone: DK-Hostmaster who has the responsibility for all .dk, should have the DNSSEC in place.

- web host: In this layer, the specific webhosting should have their own DNSSEC digital key generator.

With this method we will ensure the DNS from top to bottom. If one or more of these zones doesn't have DNSSEC, we will be a vulnerable, and hackers can easy cheat us as mentioned previously.

2.4 BIND DNS Server

2.4.1 Installation

Installation of the *BIND DNS* is done easily with Ubuntu's package manager *apt*. The following command does the job of installation:

```
sudo apt-get install bind9
```

To check the installation use a CLI tool called *dig*. If this is not available, install it in the same manner as *apt*, with the command:

```
sudo apt-get install dnsutil
```

When *dig* is available, check the installation of *BIND*. This is done by checking the loop-back interface on *127.0.0.1*.

```
dig -x 127.0.0.1
```

Which, if successful, should give an output looking like:

```
;; Query time:  3 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
```
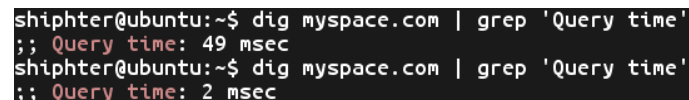
2.4.2 Configuration

The *BIND* server can be configured in a wide variety of ways, but in this report, the focus will be configuring it as a *Caching name server* and a *Forwarder*. Configuration files are located at */etc/BIND*, with the main configuration files being:

/etc/bind/named.conf.local
/etc/bind/named.conf.options
/etc/bind/named.conf

2.4.2.1 Caching name server

Using *BIND* as a *Caching name server* reduces DNS query time, since the server stores answers to name queries and only forwards the query if it does not already know the answer. By default, caching is enabled on the *BIND* server. To test this functionality a previously unvisited site is checked with *dig*, twice.

Figure 2.4: Caching name server improving query time

As expected it is found that the caching functionality vastly improves query time; see figure 2.4.

2.4.2.2 Forwarding

Forwarding is used when *BIND* cannot respond to the DNS request with the data from its cache. This is configured in */etc/bind/named.conf.options*. Here, the DNS servers that *BIND* should forward to are added in the following manner:

```
//Forwarding to Google DNS
forwarders {8.8.4.4;};
```

The reasoning behind choosing Google's public DNS is described in the next section.

2.4.3 Analyzing public DNS servers

In the quest of finding the best public DNS server, Google's tool *namebench* is used. The results looks as in figure 2.5 and 2.6.

From this, we find Google to be the best alternative, since this is both public, fast, and secure.
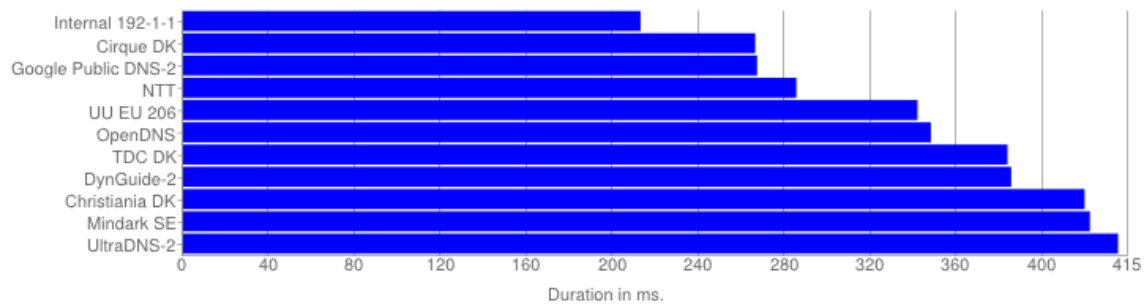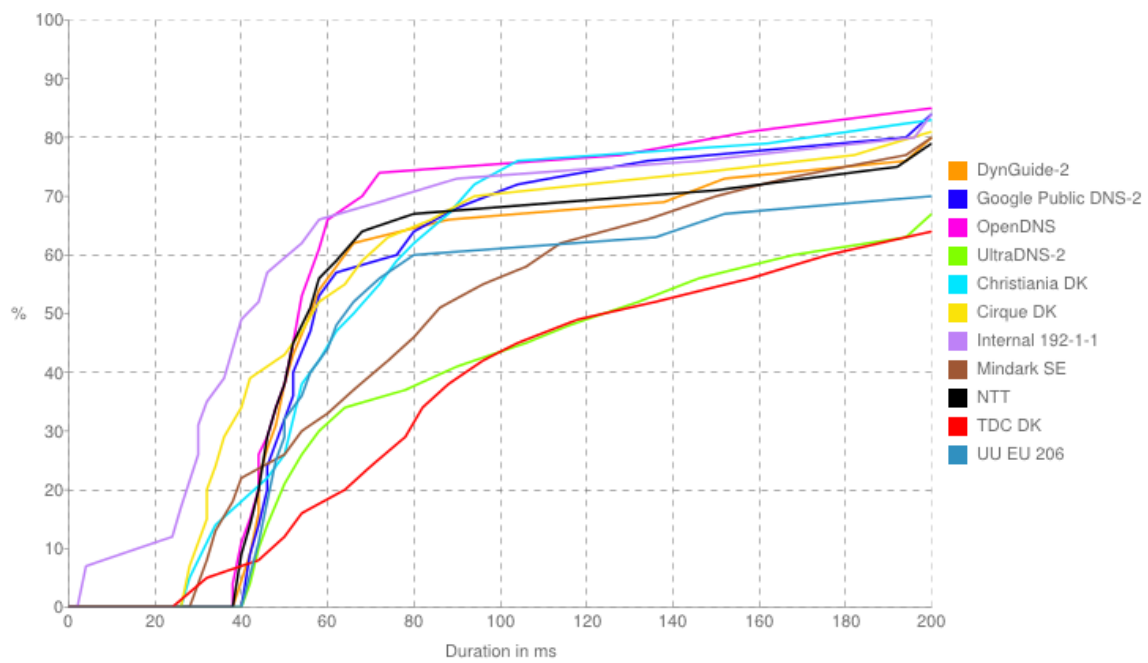
Figure 2.5: Google Namebench - Meantime response

Figure 2.6: Google Namebench - Response distribution the first 200 ms

Chapter 3

Prototype

3.1 Analysis

A public school wishes to host its own DNS, with the following functionalities:

- Caching Name Server.
- Forwarding to OpenDNS.

This is achieved solely by the use of BIND as a private network DNS Server. The schools network setup therefore will be as illustrated in figure 3.1.
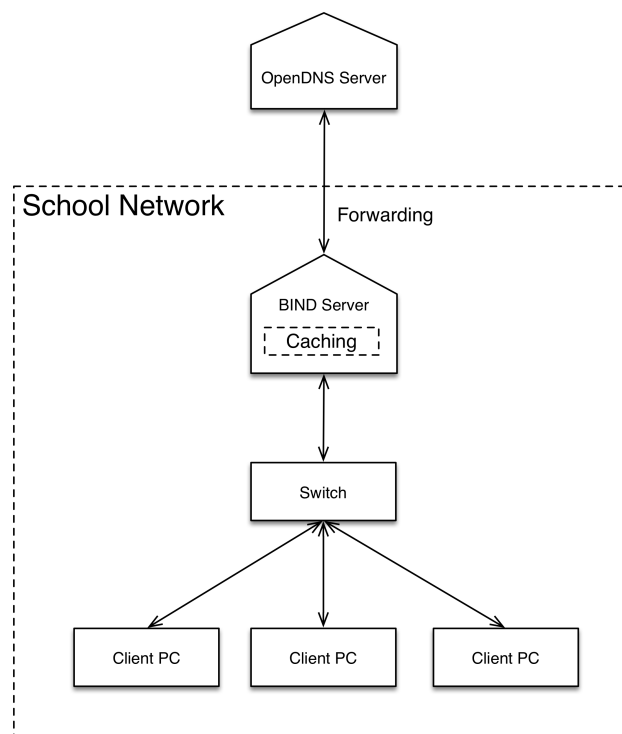

Figure 3.1: School network with BIND server.

3.2 Implementation

To host the DNS, BIND is used. A general guide to setup BIND is found in section 2.4.

3.2.1 Caching

Since the caching name server functionality is activated by default, this feature needs no further setup.
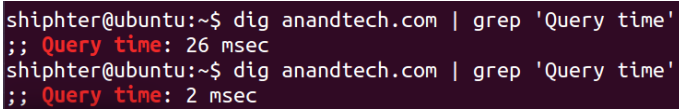
3.2.2 Forwarding

The only thing to change in the forwarding setup process is changing the use of Google's DNS to OpenDNS instead. In */etc/bind/named.conf.options* OpenDNS' address is inserted like this:

```
//Forwarding to OpenDNS
forwarders {208.67.222.222; 208.67.220.220;};
```

3.3 Test

3.3.1 Caching

To test this functionality a previously unvisited site is checked with dig, twice.

Figure 3.2: Prototype testing of Caching Name Server.

The drastic performance improvement indicates that caching functionality is fully functional.

3.3.2 Forwarding

Chapter 4

Conclusion

4.1 Conclusion

The important aspects of the project, regarding Domain Name Service has been explored both theoretically and practically. The theoretically part revolves around the fundamentals of DNS to support the practical solution, which consist of installing and setup a caching name server, using BIND software.

To further enhance the understanding of a caching name server and its features in regard to BIND, a prototype was made from a given case.

4.2 Discussion

In the beginning, each member of the group had the very same task; to complete the exercises that was given. This would give each member a basic understanding of DNS, and the overall perspective of what is needed to be accomplished to fulfill the report. To further enhance each members knowledge different subjects regarding DNS was delegated to each member.

The report is meant to cover the important aspects of DNS, upon reading the report to reevaluate each members knowledge. The prototype and the chapter regarding the conclusion were written together by the group to centralize the important aspects of the report.

4.3 Perspectives

To see other perspectives of the technology, a lot of advanced features of BIND can be explored. Features worth to be mentioned could be[1]:

- Split DNS

- Journal files

- Support IPv6

Our prototype contains the default implementation of BIND with no further features enabled. If our client, in our case, wants a private network (intranet) hidden from the public, we could enable a feature called Split DNS.

Basically what Split DNS does is to hide "internal" DNS information from "external" clients from the internet.

---

[1]More features can be found here: `http://www.bind9.net/manual/bind/9.3.1/Bv9ARM.ch04.html`

Bibliography

[1] Andrew S.Tanenbaum and Maarten Van Steen *Distributed Systems, Principles and Paradigms*, 2nd edition, 2007

[2] O'Reilly, *How does Public Key Cryptography Work in DNSSEC?, Excerpt from: MCTS Self-Paced Training Kit (Exam 70-642): Configuring Windows Server 2008 Network Infrastructure*, 2nd Ed., 2011

[3] Ramaswamy Chandramouli and Scott Rose *Challenges In Securing The Domain Name System*, 2006, IEEE SECURITY & PRIVACY, vol. 4, issue 1, 2006, pp. 84-87