Domain Name System

Handed in April 25, 2013, by Team 2

Rasmus Bækgaard	10893@iha.dk
Anders Kielsholm	10749@iha.dk
Lasse Hansen	10063@iha.dk
Mia Leth Sørensen	10959@iha.dk

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

Abstract

About your reports and this template

- Use this template for your project work reports
- Substitute the template's place-holder dates and titles with appropriate ones
- Place-holders are marked with square brackets, i.e. [place-holder]
- For each report, you hand in both a tex and a pdf file
- The report should be 10-15 pages in total and it must be written in English

About the abstract, i.e. the current section

- An abstract is a brief summary of the report that helps the reader quickly ascertain the report's purpose. The abstract should be approximately half a page.

Contents

Chapter 1

Introduction

The course on object oriented network communication is concerning distributed systems which consists of multiple computers that communicate through a computer network. In order to communicate with each other the computers need to be able to connect via an IP address. The translation/resolution from logical names to IP addresses is handled by a Domain Name Server.

This report will cover the subject of the Domain Name Server; the fundamentals about this technique, how it is working and the security issues associated with it.

Chapter 2 contains a theoretical description of the Domain Name Server including screen shots and code examples.

Chapter 3 contains a practical example of an implementation of a Domain Name Server and how it can be used to filter certain internet sites and get faster response by using a caching domain name server.

Chapter 2

Domain Name System

Domain Name System, DNS, is used to find IP-addresses from a logical name using the concept of the Host Lookup Table. The Host Lookup Table, HLT, was a file placed on every computer connected to the network which contained the IP-address and the logical name for each computer connected [1, History section].

When the HLT was updated, all computers needed to add the address which, due to the expansion of connecting computers to the network, became an obstacle and hindrance for the flow. To solve this the DNS system was invented in 1982 [2, History section].

DNS is like a big telephone book which everyone uses to lookup IP-addresses from logical names, through an address record (A record) [3, p. 210], rather than everyone keeping track of all connected addresses.

2.1 DNS fundamentals

To find the computer's host name the command `hostname` will show the logical human readable name for the computer. The hostname will be shown on the list of connected computers on a network.

In Linux the command `nm-tool` will access the NetworkManager Tool and among others, show the IP-address, MAC-address, connection state and DNS-server for the computer. This is shown in Figure 2.1[1].
You can run a similar command in Windows; `ipconfig /all` to access IP-address, DNS-server, MAC-Address ect. shown in Figure 2.2.

[1] Note that this is on a virtual machine which do not show as much as a native Linux machine will

```
limro@ubuntu:~$ nm-tool

NetworkManager Tool

State: connected (global)

- Device: eth0  [Wired connection 1] ----
------------
  Type:              Wired
  Driver:            vmxnet
  State:             connected
  Default:           yes
  HW Address:        00:0C:29:D5:8E:55

  Capabilities:
    Carrier Detect:  yes
    Speed:           1000 Mb/s

  Wired Properties
    Carrier:         on

  IPv4 Settings:
    Address:         192.168.92.128
    Prefix:          24 (255.255.255.0)
    Gateway:         192.168.92.2

    DNS:             192.168.92.2
```

Figure 2.1: Use of the command `nm-tool`

```
C:\Users\Becks>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : Becks-PC
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : iha.dk
   System Quarantine State . . . . . : Not Restricted


Wireless LAN adapter Wireless Network Connection:

   Connection-specific DNS Suffix  . : iha.dk
   Description . . . . . . . . . . . : Broadcom 4313 802.11b/g/n
   Physical Address. . . . . . . . . : E0-2A-82-A7-2E-43
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::29e7:43ec:7fc0:49c4%14(Preferred)
   IPv4 Address. . . . . . . . . . . : 10.193.2.193(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : 16. april 2013 12:05:07
   Lease Expires . . . . . . . . . . : 17. april 2013 00:55:45
   Default Gateway . . . . . . . . . : 10.193.2.1
   DHCP Server . . . . . . . . . . . : 10.88.1.95
   DHCPv6 IAID . . . . . . . . . . . : 383789698
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-18-F8-51-A0-64-31-50-0E-14-76

   DNS Servers . . . . . . . . . . . : 10.20.255.36
                                       10.20.255.33
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

Figure 2.2: Use of the command `ipconfig /all`

To detect an IP-address and determine the latency to the server use the `ping` command. `ping` will target either a webserver name to get the IP-address or target the IP-address directly without asking the DNS-server. This is shown on figure 2.3

```
limro@ubuntu:~$ ping -c 3 www.google.com
PING www.google.com (173.194.65.105) 56(84) bytes of data.
64 bytes from ee-in-f105.1e100.net (173.194.65.105): icmp_req=1 ttl=128 time=24.7 ms
64 bytes from ee-in-f105.1e100.net (173.194.65.105): icmp_req=2 ttl=128 time=30.1 ms
64 bytes from ee-in-f105.1e100.net (173.194.65.105): icmp_req=3 ttl=128 time=36.3 ms

--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 24.746/30.437/36.375/4.754 ms
limro@ubuntu:~$ ping -c 3 173.194.65.105
PING 173.194.65.105 (173.194.65.105) 56(84) bytes of data.
64 bytes from 173.194.65.105: icmp_req=1 ttl=128 time=25.4 ms
64 bytes from 173.194.65.105: icmp_req=2 ttl=128 time=24.7 ms
64 bytes from 173.194.65.105: icmp_req=3 ttl=128 time=24.0 ms

--- 173.194.65.105 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 24.080/24.764/25.468/0.566 ms
```

Figure 2.3: Use of the command `ping -c 3 www.google.com`

Before makeing a lookup at the DNS server the system will check local HLT file, located in */etc/hosts*, to see if any redirects or A records are listed. Redirections are created by typing the static IP-address followed by the new logical name as shown in Code snippet 2.1

Code snippet

```
# Redirections
212.130.55.139 vejr #Resolve vejr to "www.dmi.dk"
159.20.6.38 nyhed #Resolve nyhed to "www.dr.dk"
157.55.46.241 mail #Resolve mail to "www.hotmail.com"
```

When requesting a webserver through a DNS, the root servers first redirect to the top-level domain server, TLD, which contains the '.com', '.net', '.dk' etc [3, p. 192]. From here the domain server can return the IP-address for the name server to which the client can connect. This will then (most likely) recursively or (less likely) iterative[2] return the specified IP-address to the client.

Each DNS server is responsible for looking up domains in nonoverlapping parts called 'zones'. A zone is implemented by a separate name server [3, p. 202-205].[3]

When looking for a host name there can be multiple units with the same logical name. However it is possible to specify a name for a unit to unambiguously unique with a '.' (dot) at the end of the logical address which is called a fully qualified domain name (FQDN) [4].
Multiple computers could have the name *myComputer* but if looking for a computer on a network called *work.net.* there can only be one computer with the name *myComputer.work.net.*

[2] See section 2.2, Name resolution
[3] FiXme Note: Find eksempel

2.2 Name resolution

When looking up an IP-address it can be resolved in two different ways. In iterative name resolution the root DNS server return the IP-address for the a DNS server which contains information of the requested country code. Afterwards the client will ask this DNS server for the IP-address and so on until the client the requested IP-address. This DNS query/response transaction type of resolution requires the client involvement for each request to a DNS server which leads to lower performance cost for the DNS servers [3, p. 205-209].

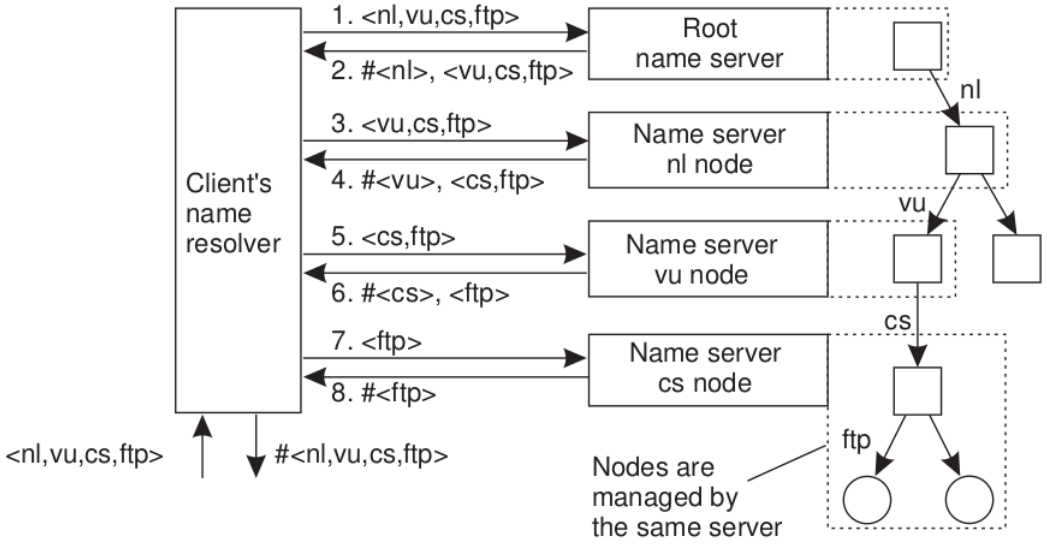

Figure 2.4: Iterative request for an IP-address

In recursive name resolution the root DNS server will ask the country coded DNS server for the IP-address of the website. The country coded DNS server ask the another DNS server which contains specific information of the domain and so on, until the website is identified. The IP-address is returned recursively to the root DNS server and back to the client. Caching the returned IP-address can lower the performance cost drastically, since a lookup is unnecessary for the same request next time.

Therefore the client is only involved when asking for and obtaining the IP-address which reduce performance cost for the client [3, p. 205-209].

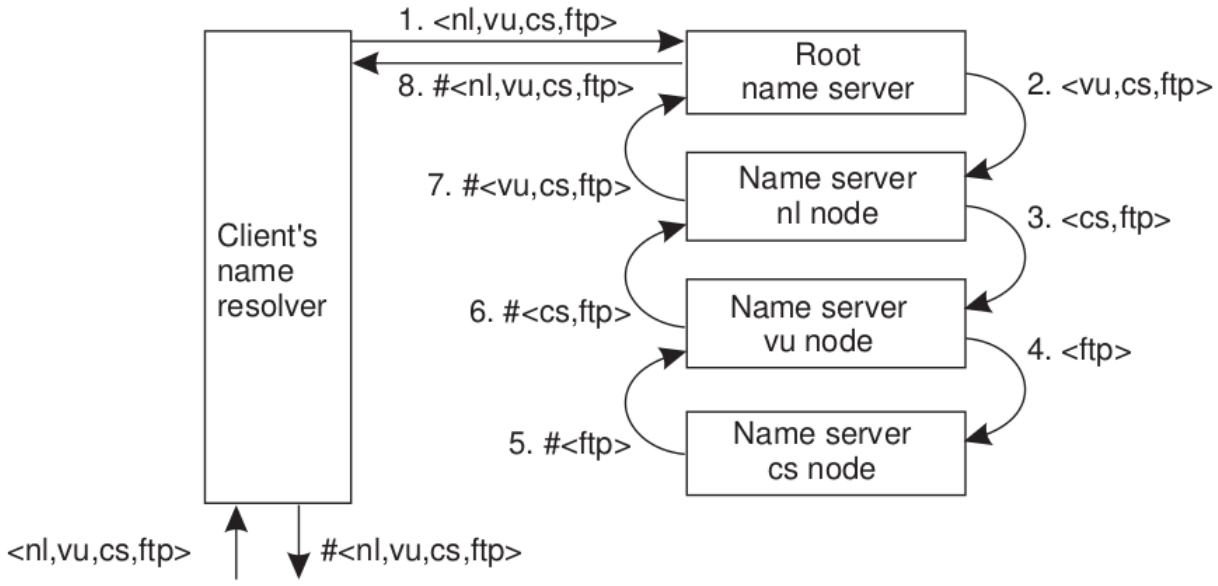

Figure 2.5: Iterative request for an IP-address

2.3 DNS security extentions

Due to security issues various governments, research organizations and others have developed a specification and associated protocol called DNS Security extensions (DNSSEC) which protects DNS querty/response transactions [**?**, p. 84].

Two main security threats exist for DNS in the context of query/ response transactions. Attackers can

- spoof authoritative name servers responding to DNS queries and alter DNS responses
- alter the DNS responses stored in caching name servers.

[**?**, p. 84]

DNSSEC was designed to protect the users from obtaining corrupted DNS data. It contains a set of extensions to DNS which provide origin authentication of DNS data, authenticated denial of existence, and data integrity. If a DNS server uses DNSSEC it is marked as a *signed zone*.

A signed zone is a DNS server zone which includes a digital signature for all content it returns. It verifies that the underlying responses have requested resource records, special resource records that carry the digital signatures associated with the requested resources and it contains a DNSKey which include a public key which can verify the signature.

Using signed zones as authentication requires a DNSSEC-aware caching name server which start from a trusted public key stored within it self, a *trust anchor*. This establish a chain of trusted DNS servers with DNSSEC implemented to the public key of the zone. It is also possible to use a *trust achor list* which contains a list over trusted anchors.

Chapter 3

Prototype: School DNS

A school want to use a DNS server to filter certain internet sites from students and also have the opportunity to get a faster response from the name server.

3.1 Solution

To achieve the school's request, a BIND server could be set up. BIND is a open source implementation of the DNS protocols and is the most used DNS server software. One of the advantages with BIND, is that it supports both Windows, Mac and Linux. BIND acts like a caching server, where it stores answers to name queries and this results in reduced time of future queries to the same server.

When a requested hostname needs to be looked up, the local cache is the first place to be checked. If the hostname is not found there, the next DNS server is asked. This might be your ISP or even a server on a "higher" level. It is also possible via BIND to forward your requests to a public DNS, meaning the server will be asked, if no other server before that (e.g. your ISP's DNS) have the hostname in it's cache.

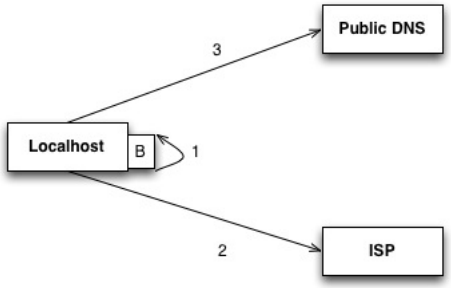

Figure 3.1: System overview

3.1.1 Setup BIND Server

To install a BIND server on Linux type in "sudo apt-get install bind[9]". This will install version 9 of the BIND server software. To check if installation if successful type `named -v` and if it is successful, it will show `BIND 9.8.1-P1`. For testing purpose, *dnsutils* have been used - and this can be used to see Query time for the DNS lookup with the commannd `dig -x [IP-address]`.

3.1.2 Forward to public DNS

If the school whats to forward their requests to a public DNS, e.g. OpenDNS, they have to do a bit of research on finding the optimal solution for them. There is a lot of public DNS servers, where some servers are good at filtering bad hostnames, but are in the same time not always the fastest. In this section tests are made to find the fastest response time.

To find an optimal solution, Google's Test Bench (GTB) have been used. In this case GTB looked up around 4500 servers and tested them all to find the fastest server in average.

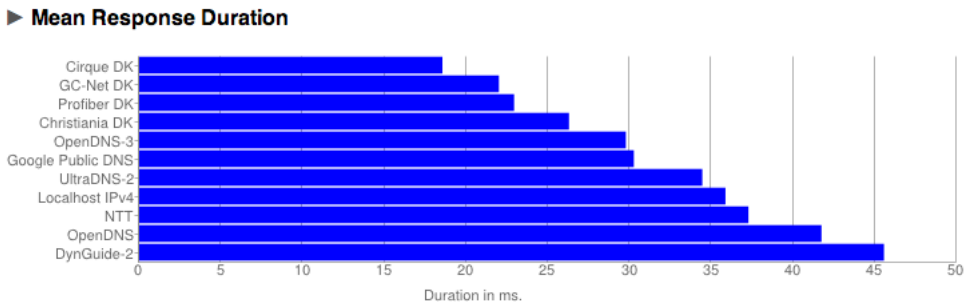

Figure 3.2: Output from Google Test Bench test

Forwarding - Cirquie.DK

Site	First test (ms)	Second test (ms)
Ubuntu.com	391	381
Bt.dk	356	906
Iha.dk	375	240
Facebook.com	354	207
Wikipedia.org	375	442

Forwarding - OpenDNS

Site	First test (ms)	Second test (ms)
Ubuntu.com	355	352
Bt.dk	792	436
Iha.dk	334	117
Facebook.com	184	279
Wikipedia.org	153	115

Table 3.1: Forwarding with Cirquie.DK and OpenDNS

The test in Figure 3.2 and Table 3.1 shows, that Cirque DK have the lowest mean response time with 18ms, and the default DNS have a mean response time on 36 ms. One of the more popular public DNS is openDNS, which is a little faster than the default DNS with 29 ms. To test if one of the public DNS is faster, a manual test with `dig -x` have been made with 5 different internet sites and their given response time.

To test the given servers, the file located at */etc/bind/named* have to be edited with the address of the DNS server it shall forward to.

It is hard to make a final conclusion based on our test, since the response times are unstable (e.g. Cirquie.DK with a response time on 356 ms on Bt.dk in the first test, and 906 ms in the second test and both test was under the same conditions), which is why Google Test Bench is a good tool to use. However, we can conclude that even if you forward to the one tested to be the fastest public DNS, you cannot be sure it is the fastest every single time, since it might be busy.

3.1.3 Filtering

A DNS server can also be used as a simple filter, in the way of not providing the IP for harmful or illegal hostnames. Some public DNS servers have this filter build-in, which means you in a simple way can archive a bit of security.

In the same way, it is also possible to block specific hostnames with BIND, which can be useful if you want your filter to be more specialized. This can be useful if a school wants to block for e.g. Facebook or Ekstrabladet.dk.

The problem about the filter is if the users look up the IP directly. In that case, the DNS server will not be used and the filter will therefore not be used either and other methods have to be used.

3.1.4 Discussion

There is a lot of ways of finding a solution for the school. To retrieve the caching functionality, BIND is simple and easy to set up and use. If wanted, it can furthermore be specialized in the way of setting up a filter or forwarding requests to a public DNS server.

BIND is a simple solution, but if better security or caching is wanted, a custom solution is needed. However, for a small school this might not be needed, and therefore BIND would be the suggested choice.

Chapter 4

Conclusion

Approximately 1-2 pages covering conclusion, discussion, and perspectives.

4.1 Conclusion

Conlude on your investigations.

4.2 Discussion

Discuss your project work.

4.3 Perspectives

What are the perspectives on the technology and your prototype?

Bibliography

[1] Wikipedia, "Hosts (file)," April 2013. Accessed 19-04-13, URL: http://en.wikipedia.org/wiki/Hosts_(file).

[2] Wikipedia, "Domain name system," April 2013. Accessed 19-04-13, URL: http://en.wikipedia.org/wiki/Domain_Name_System.

[3] A. S. Tanenbaum and M. Van Steen, "Distributed systems principles and paradigms," *Network*, vol. 4, p. 20, 2004.

[4] Wikipedia, "Fully qualified domain name," April 2013. Accessed 19-04-13, URL: http://en.wikipedia.org/wiki/Fully_qualified_domain_name.