

# **ITONK Project**

**(Please include DNS, DDS, or RMI in the title)**

**Handed in [Month Day], [Year], by team no. [n]**

Stephan Thordal Larsen	11072@iha.dk
Cong Thanh Dao	20073860@iha.dk
Kasper Duong	08754@iha.dk
[Name 4]	[Email address]
[Name 5]	[Email address]

**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 reports
- Substitute the [place-holders] with your text. Do not keep the square brackets
- You are free to add chapters and sections
- Hand in the reports as pdf files - one pdf per report.
- The report should be around 15 pages in total and it must be written in English

Why  $\LaTeX$ ?

- I want you to try Latex as this is a very powerful and widely used document preparation system that is de facto standard in, e.g. mathematics, physics, computer science and engineering.

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

<b>Abstract</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
<b>2 Technologies - DNS, DDS or, RMI</b>	<b>4</b>
2.1 DNS Fundamentals . . . . .	4
2.2 Name Resolution . . . . .	7
2.3 DNS Security Extension . . . . .	8
2.4 Bind DNS Server . . . . .	9
<b>3 Prototype</b>	<b>11</b>
<b>4 Conclusion</b>	<b>12</b>
4.1 Conclusion . . . . .	12
4.2 Discussion . . . . .	12
4.3 Perspectives . . . . .	12
<b>Bibliography</b>	<b>13</b>

## Chapter 1

### Introduction

Approximately 1 page introduction that addresses the following

1. What the report is about
2. Why the report is relevant
3. How the rest of the report is structured

These are test citations to example bibliography entries number one [1] and two [2]. You should have **at least 3 references** to books and/or papers, i.e. web pages excluded. Of course, you should not refer to these test entries unless you in fact use them.

## Chapter 2

# Technologies - DNS, DDS or, RMI

This chapter should contain an in-depth description of the technology, i.e. DNS, DDS, or RMI. You should at least address

- The purpose of the technology
- Technology alternatives
- Downloading, installing, configuring, and employing the technology

### 2.1 DNS Fundamentals

The DNS - Domain name service is used to resolve host names, so it is possible to search for a name, instead of an ip address. This ensures that you can make a search for ea. `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:

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 - Windows

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

```
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*

Now, we want to demonstrate in Windows about pingg 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](http://www.dr.dk) and then type in the IP-address in a webbrowser - it will show the same result:

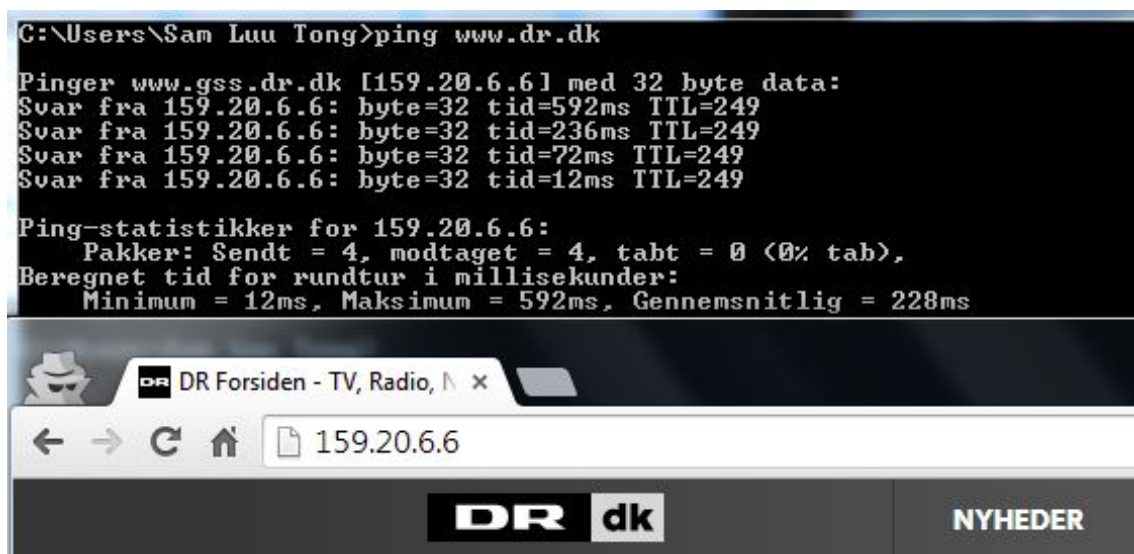


Figure 2.2: Screenshot of pingg [www.dr.dk](http://www.dr.dk)

So whether we type [www.dr.dk](http://www.dr.dk) or we type 159.20.6.6 doesn't make a difference - only that [www.dr.dk](http://www.dr.dk) probably is easier to remember!

### 2.1.3 Host Lookup Table - Background and demonstration

#### 2.1.3.1 The Background

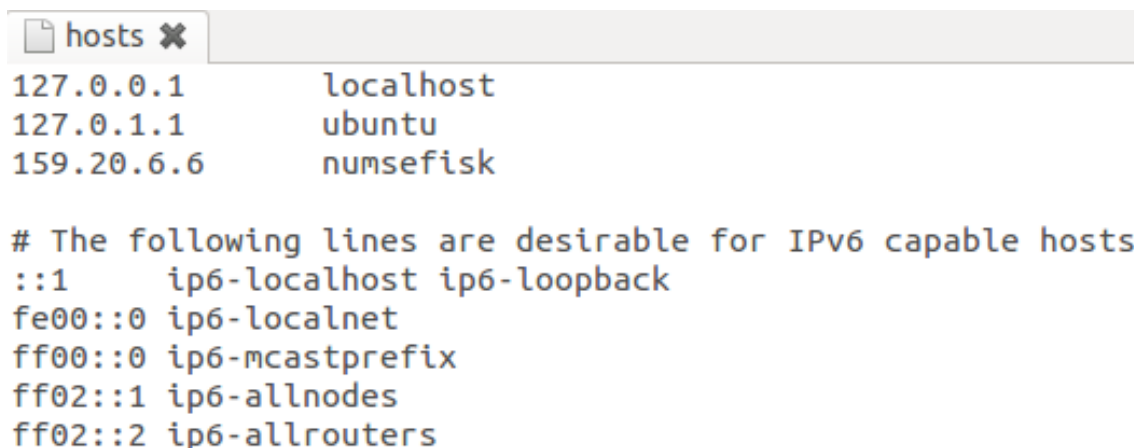
Before DNS was fully deployed, a woman, **Peggy Karp**, created a lookup table that mapped all of the network resources in one text formatted file. This formatted file was a simple ASCII text file called "HOSTS.TXT", and the table contained all of the hostnames and their related IP addresses.

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:



```
hosts x
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.<sup>1</sup> If you look at an internet address, it is made of the `www.second-level-domain.com`, the `.com` is the top level domain. The TLD defines in which TLD the router shall look for the second-level-domain name. The `.com` TLD is a generic TLD (gTLD), which states for which purpose they are to be used. Also there do exist country code TLD's (ccTLD's) like `.dk`, `.jp` or `.de`.

#### **2.1.5 Fully Qualified Domain Name**

A Domain Name can be either fully qualified(FQDN) or not. The definition of a FQDN is, that it is unambiguous, 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 (<http://www.dns-sd.org./TrailingDotsInDomainNames.html> ).

#### **2.1.6 DNS zone and records**

when a domain name is looked up, then a DNS zone returns a record, possible 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.

A DNS zone is a portion off a domain name system, just like the areal code in a post district. The zone has been delegated administrative responsibility for the DNS.

sources: [1]

### **2.2 Name Resolution**

When we want to access a specific host via a web browser, 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 its 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 request to a DNS server, which will 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.

So for instance when a client request for a host name, `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 then choose one of the `.COM` gTLD Server and then query for the `example.com`, and if found it will reply back and the DNS Server will respond back to the client with the IP address that corresponds to the host name.

---

<sup>1</sup><https://archive.icann.org/en/tlds/>



### **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 make it vulnerable for hacking. There are several ways to integrate security directly in the DNS, One of them is DNSSEC. DNSSEC attempts to add security, while maintaining backwards compatibility.

### **2.3.1 Threads**

There are several threads in DNS 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.

DNSSEC is a digital signature, which is set on the data to ensure that they are valid. 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. It therefore 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, taking DNSSEC in use at each stage of the DNS process.

DNSSEC incorporates a chain of digital signatures in the DNS hierarchy, with each level owns its own signature generating key. This means that the root zone signing a key .dk-zone and that .dk-zone signing key to iha.dk name server. ICANN, who controls the Root Server has DNSSEC in place, DK Hostmaster that controls all .dk domain names

have DNSSEC in place, and many web hosts will also provide their users with DNSSEC, so the various links in the lookup process ensures each other.

## 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)
```

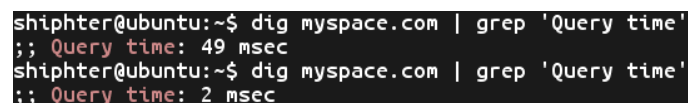
### 2.4.2 Configuration

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*, two times.



```
shiphter@ubuntu:~$ dig myspace.com | grep 'Query time'
;; Query time: 49 msec
shiphter@ubuntu:~$ dig myspace.com | grep 'Query time'
;; Query time: 2 msec
```

Figure 2.4: Caching name server improving query time

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

#### 2.4.2.2 Forwarding

Forwarding is used when *Bind* can not respond to the DNS request with the data from it's 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 follows:

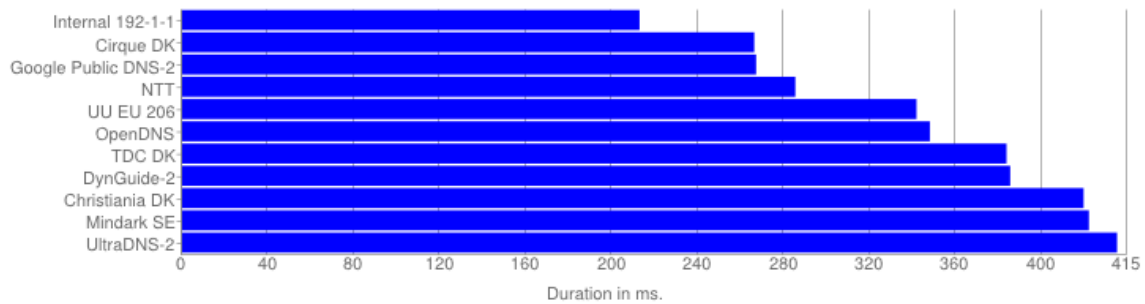


Figure 2.5: Google Namebench - Meantime response

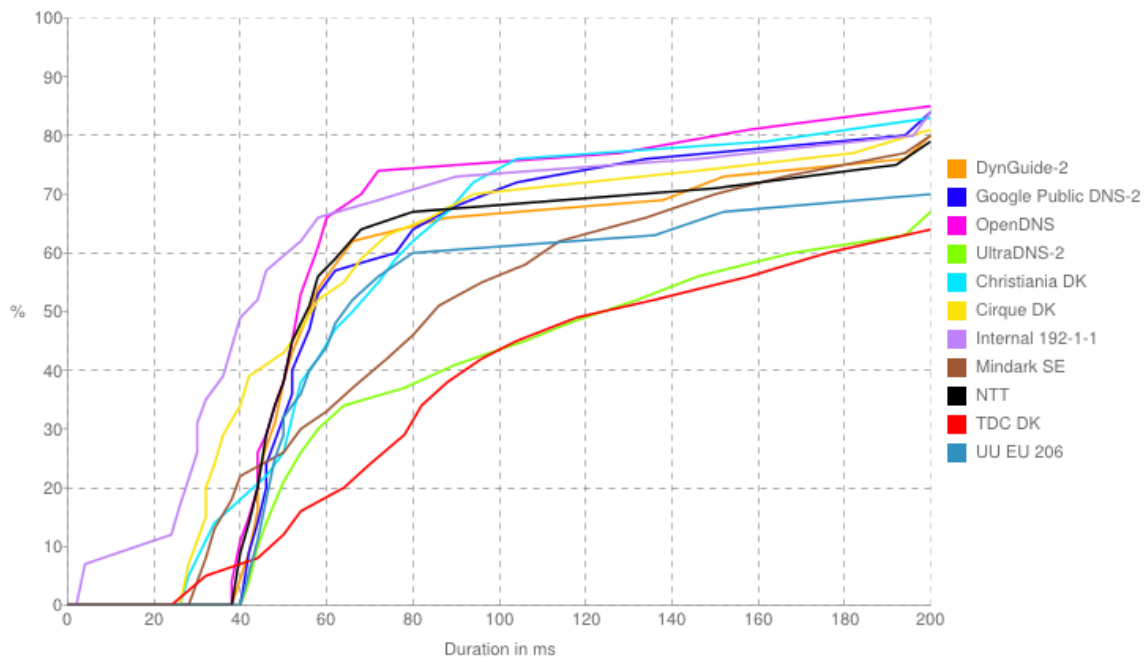


Figure 2.6: Google Namebench - Response distribution the first 200 ms

From this, we find Google to be the best alternative, since this is both public, fast and secure.

## Chapter 3

# Prototype

This chapter should contain an in-depth description of prototyping with the technology, i.e. DNS, DDS, or RMI. That is, you analyze, design, implement, and test

- a very limited, but functional prototype that utilizes the technology under consideration.

You define your own prototype and the context in which it should function; the list below is for your inspiration.

- Domain Name System: A public school or a medium sized company would like to host their own DNS and/or forward requests to OpenDNS.
- Data Distribution Service: A hospital or a production factory would like to employ Connect DDS to distribute mission critical data.
- Java Remote Method Invocation: A company is setting up facilities, e.g. parcel or luggage sorters, abroad and would like to be able to access back-end methods and data at home.

In your analysis you should at least address and/or include

- Overall diagram and description of the prototype
- Relevance of the technology under consideration to your prototype
- How the technology is included in your prototype
- Definition of a small set of realistic use-cases and related functional requirements

The design, implementation, and test should at least address and/or include:

- Diagrams, e.g. UML, supplemented with code snippets of most important parts
- Test and evaluation of your system: Does it work as intended?
- Evaluation of the prototype and the technology employment as a whole

## **Chapter 4**

### **Conclusion**

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

#### **4.1 Conclusion**

Conclude 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] R.L. Graham, D.E. Knuth, and O. Patashnik, *Concrete mathematics*, Addison-Wesley, Reading, MA, 1989.
- [2] H. Simpson, *Proof of the Riemann Hypothesis*, preprint (2003), available at <http://www.math.drofnats.edu/riemann.ps>.